Ára: 30 Ft

μ mikro számítógép magazin

REPREZENTÁCIÓ 1987/7

Szki PRONET LOKÁLIS HÁLÓZAT

HARDVER

— A PROPER—16 (IBM kompatibilis) számítógépcsaládból összeállítható hálózat kétirányú busz szervezésű
— A hálózat kiépítéséhez PRONET lokális hálózati kártya is szükséges
— A hálózat állomásai közös átviteli közegre kapcsolódnak
— A hálózatba újabb állomások könnyen bekapcsolhatók

Főbb műszaki jellemzők

- adatátviteli sebesség: 1 Mbit/s
- adatátviteli közeg: csavart érpárú vagy telefonkábel
- a főkábel szegmens hossza: maximum 300 m, repeater-rel 1200 m
- Az állomások elméleti maximális száma: 255. A hatékonyan üzemeltethető állomások száma alkalmazásfüggő: a hálózatokba kapcsolt erőforrások és a felhasználás jellege együttesen határozzák meg.

SZOFTVER

— PROPOS V.3.30 vagy ezzel kompatibilis operációs rendszer
— PRONET 3.0 hálózati szoftver
  - IBM PC NETWORK program kompatibilis
  - PRONET BIOS (IBM NETBIOS EMULÁCIÓ)
  - FILE SERVER
  - PRINTER SPOOL SERVER
  - MESSAGE SERVER
  - PRONET—BASE a dBASE, CLIPPER hálózati kiegészítése
  - Elektronikus Posta
— A hálózat szolgáltatásainak elérése külső parancsok segítségével, operátori konzolról vagy programból (felhasználói interfész) biztosított

Információ:

Sci-L

Számítástechnikai Informatikai Fejlesztő Leányvállalat
1011 Budapest I., Iskola u. 10.

SCITEL Számítástechnikai Fejlesztő Leasing Leányvállalat
1015 Budapest I., Donáti u. 35—45.

5. ÉVFOLYAM
1987/7. SZÁM

μ mikro számítógép magazin

A NEUMANN JÁNOS SZÁMÍTÓGÉPTUDOMÁNYI TÁRSASÁG ÉS A KISZ KÖZPONTI BIZOTTSÁG LAPJA

A kiadvány a Tudományszervezési és Informatikai Intézettel együttműködve készül

A szerkesztőbizottság vezetője:
Kovács Győző

E számunkat szerkesztették:

Bakos Tamás
(programozástechnika)

Broczkó Péter
(hírek)

Kovács Győző
(levelezés)

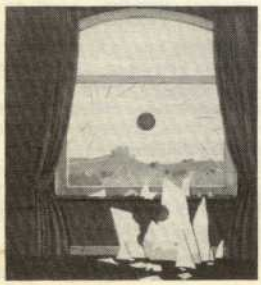
Lindner László
(sakkprogramozás)

Petróczy Judit
(könyvek)

Simonyi Endre
(klub)

Varga András
(iskola-számítógép)

Címképünk:
Ramocsai Imri munkája
(RENÉ MAGRITTE után)



Felelős szerkesztő:
Könyves Tóth Pál

Szerkesztőség:
1027 Budapest, Fő u. 68.
Telefon: 154-250

Levélcím:
1371 Budapest
Pf. 433.

Kiadja az Ifjúsági Lap- és Könyvkiadó Vállalat

Felelős kiadó:
dr. Petrus György
igazgató

Kiadóhivatal:
1065 Budapest, Révay u. 16.
Telefon: 116-660

Terjeszti a Magyar Posta
Előfizethető a hírlapkézbesítő hivataloknál
és a Posta Hírlapelőfizetési és Lapellátási Irodáján
(1900 Budapest V., József nádor tér 1.)
vagy átutalással a 215—96 162 pénzforgalmi jelzőszámra.

Megjelenik havonta
Egy szám ára 30,— Ft
Előfizetési díj:
egy évre 360,— Ft
fél évre 180,— Ft
Külföldön terjeszti a Kultúra,
1389 Budapest, pf. 149.
és a Magyar Média
1932 Budapest, pf. 279.
86—0253

Szikra Lapnyomda
Budapest (87-0803)
Felelős vezető:
Csöndes Zoltán vezérigazgató

INDEX: 25 629
ISSN 0236-6088

Tartalom

Panaszkodunk 2
Adok-veszek-cserélek 8
OKTA-TOTÓ 9
Atari kontra Commodore 64 10
Pályázat 20
Ipari park a műegyetem mellett 21
Vigyázat! Tolvaj! 22
Mesterséges értelem III. 23
Az adattípus fejlődése I. 27
Spectrum a laborban 29
μINFORM 36
Olvastunk . . . 38
Meditáció 41
Aprócska 45
Fórum 47

ISKOLA — SZÁMÍTÓGÉP

„Memory" suli 3
Gépi kódban 5

DIÁKROVAT

Nyújtott karakterek 6
Finom kiíratás Primóra 6
Grafika 7
Három rutin 8

PROGRAMOZÁSTECHNIKA

BASIC és gépi kód 14
Z80 programok haladóknak 15
Teknősbéka-grafika III. 16
Adatbeolvasás Pascalban 17

μPROGRAMOK

Hasznos programok az URS függvény alapján 30
„Piszok" ellen sortörlő 31
Relatív adattárolás 32
Színváltás 32

μKLUB

Integrált szoftver 34
A VC—1541 lemezegység furcsa titkai 35
Adom a magyarázatot! 36

SAKKPROGRAMOZÁS

Bitek és figurák 42
A mikrogépek 6. világbajnoksága 43

AZ OLVASÓ ÍRJA 46

KÖNYVEK 47

HÍREK, ÉRDEKESSÉGEK 48

Panaszkodunk

„Nem találunk tehát semmilyen ésszerű alapot ahhoz, hogy a gépeknek tudatot tulajdonítsunk. Helytelen tehát a gépekben lévő folyamatokat az agyelvű működésével vagy „gondolkodással" magyarázni. Tulajdoníthatunk a gondolkodásnak is hasonló folyamatokat (mint amilyenek a gépekben mennek végbe), azonban ez több, mégpedig annyival több, amennyivel az élő szervezet több a gépnél"

(Dr. Heinz Zemanek: Információelmélet I. 1956. Fordította: Nozdroviczky László)

Az elmúlt hónapokban több számítóközpont-vezetővel is összefutottam, akik — mintha összebeszéltek volna — arról panaszkodtak, hogy egyre nehezebb a nagyszámítógépekre felhasználókat találni, a számítóközpontok jó ha két műszakban dolgoznak, akkor sincsenek a teljesítményük végső határáig kihasználva. Abban is egyezett a véleményük, hogy ennek a problémának a gyökere a mikroszámítógépek gyors elterjedésében keresendő, a bajok eredetét a hetvenes évek végére tehetjük — akkor jelentek meg ugyanis a piacon az első személyi számítógépek. Egészen addig nagy kereslete volt a számítógép-kapacitásnak, a számítóközpontok legfeljebb abban versenyeztek, hogy ki kérjen többet egy óra gépidőért.

Akkoriban a számítóközpontok jó része batch üzemben működött, és óriási eredménynek számított, ha egy-egy program átlagos fordulási idejét két napról egy napra lehetett mérsékelni.

Abban a szerencsés helyzetben voltam, hogy részt vehettem az első time-sharing üzemű és zömében terminálokról elérhető számítógép munkába állításában és az üzemeltetés megszervezésében. Munkatársaimmal együtt tanultam az ilyen nagy teljesítményű számítógépek üzemszervezésének technikáját: hogyan lehet a számítógép munkáját folyamatosan ellenőrizni és a kihasználását még csúcsidőben is az optimum közelében tartani. Sokat küzdöttünk, hogy a terminálok mellett ülő programozó ne érezze úgy, hogy a gép nagyon le van terhelve, a terminálokon a válaszidő néhány másodpercnél ne legyen több.

Arra is emlékszem, hogy a programozók, akik az előző gép lecseréléséig zokszó nélkül eltűrték, hogy a tesztelésre leadott programjuk naponta egyszer vagy jobb esetben kétszer kerüljön a gépre, a terminál mellett, ha a válaszidő valami miatt egy percnél hosszabb lett, azonnal telefonáltak a gépterem vezetőjének vagy nagyon sokszor nekem is, hogy tegyek rendet, mert képtelenség dolgozni, ha ilyen hosszú várakozásra kényszerülnek. Igazuk volt, egy nagygépes üzem végezze a munkáját úgy, ahogy egy ilyen rendszerben végezni kell.

Én azt hiszem, kár nosztalgiázni és a régi szép nagyszámítógépes időket emlegetni, közben pedig arra várni, hogy valami csoda történik, eltűnnek azok az „átkozott" PC-k, és akkor ismét lesz munkájuk a számítóközpontoknak.

Kíváncsiak lehetünk, mi a helyzet máshol, például a szomszédos Ausztriában. Alkalmam volt (az ADV — Arbeitsgemeinschaft für Datenverarbeitung — kongresszusán) néhány intézmény számítóközpontjának vezetőjével beszélni; volt közöttük olyan igazgató is, aki nemcsak az intézménye belső számítási munkáihoz biztosította a gépi kapacitást, hanem, az egyre csökkenő belső támogatás miatt, kiment a „szabadpiacra" is, hogy a termékét — a gépidőt — áruba bocsássa.

A helyzet ott is nagyon hasonlít arra, amit az itthoni kollégák mondtak. A számítási feladatokat a termelő, kereskedelmi, tudományos és mindenféle más intézmények zömében mikroszámítógépeken oldják meg, legtöbbször IBM PC kompatibilis gépeken. (Csak közbevetőleg, érdekességként hadd említsem meg, hogy az ADV kongresszusán az *IBM PC kompatibilis gép* kifejezést szinte sohasem hallom. Megkérdeztem, miért van ez így. Felvilágosítottak, hogy ha jobban megfigyeltem volna az előadásokat, akkor fel kellett volna, hogy tűnjön nekem egy másik fogalom gyakori használata, nevezetesen: *szabványos ipari számítógép*. Ezt mondják az IBM PC kompatibilis számítógép vagy egyszerűen IBM klón helyett, nem tartják ugyanis helyesnek, hogy az IBM-nek a szakma ilyen nagy reklámot keltsen — ingyen.)

Miután piacon vannak már a néhány MB tárolási kapacitású megamikrók, az az általános tendencia, hogy a napi számításokat mindenki az irodájában az azonnal hozzáférhető *szabványos ipari számítógépeken* oldja meg, megfordíthatatlan.

Mi legyen ilyen helyzetben a nagyszámítógépekkel, amelyek — Ausztriáról lévén szó — jóval nagyobbak és így jóval drágábbak is, mint a hazai gépek, így ha nem dolgoznak, a ráfizetés is nagyobb, mint nálunk.

Az első lehetséges megoldást, hogy panaszkodnak és igyekszenek összetörni a mikrókat, elvetették. Maradt a második megoldás: arra használják a nagygépeket, amire a *szabványos ipari számítógépek* alkalmatlanok, nevezetesen nagy tömegű adattárolásra. Már Ausztriában is egymás után jönnek létre az adatbankok, amelyek örömmel ajánlják fel szolgáltatásaikat mindenkinek, akinek információra van szüksége. Szerencsére megnőtt a vállalatok „adatéhsége" is. A szigorodó piaci viszonyok arra kényszerítik a termelőket, hogy ha valaminek a gyártásába belefognak, először nézzenek körül a piacon, gyűjtsék össze mindazokat az információkat, amelyeket csak elérhetnek (marketing). A tervezők a szabadalmi adatbankokban „turkálnak", a vegyészek a kémiai receptúrákat böngészik a Chemical Abstracts adatai között. Nem is beszélve az újságírókról, akiknek a különböző kiadók politikai, tudományos, művészeti és más adatokat kínálnak a számítógépeikről. Létrejött a videotex hálózat, ami többezer előfizetőnek, közöttük nagyon sok magánelőfizetőnek ad magas szintű számítógépes szolgáltatást, és még arra is lehetőséget nyújt, hogy bárki magán-adatbankot létesítsen a központi rendszerben.

Ezek a nagyszámítógépek persze számolnak is, nem csak az adatokat tárolják. Olyan számításokat végeznek, amelyekbe a mikrók belebuknának: például műveleteket óriási adattömbökkel, nagyméretű gazdasági modelleket számolnak. Szinte egyöntetű volt az osztrákok véleménye,hogy ma ezek a számítások a teljes üzemidőnek legfeljebb a 20%-át töltik ki, a maradék 80%-ban a gép az adatok karbantartásával, illetve adatszolgáltatással foglalkozik.

A hazai helyzetről egy alapvicc jut eszembe, biztos sokan ismerik is. Egyszer a püspök meglátogatja az egyik plébániát, az autóból kiszállva megdöbbenve észleli, hogy nem szól a harang, pedig az előírás szerint a püspököt harangszóval kell fogadni.

— Miért nem harangoztat, plébános úr? — kérdezi a püspök.

— Hát, sok oka van annak — mondja a plébános — egy: nincs harang.

Ez a vicc hozakodik elő, amikor arra gondolok, hogy egy jól működő adatbankhoz nem elég csak a nagyszámítógép: adathálózat is kellene, amelyen keresztül az adatbank szolgáltatásait el lehet érni. Tudom, hogy 13 fővonal/100 fő távbeszélősűrűség mellett — komoly adathálózat hiányában — az amúgy is túlterhelt telefonközpontoktól azt is elvárni, hogy a számítógépeket is kiszolgálják, naivitás. Ennek ellenére ma ez az egyetlen lehetőség arra, hogy a nem teljesen leterhelt nagyszámítógépek jobban ki legyenek használva, ráadásul olyan feladattal — adatszolgáltatással —, amely a népgazdaság részére közvetlen hasznot jelent.

Miután alig van nyilvános, bárki által hozzáférhető adatbank az országban, ez valószínűleg azt jelenti, hogy a hazai intézmények nem nagyon keresik az adatokat, különösen nem a számítógépen tárolt információt. Ha lenne piaci kereslet, talán lenne vállalkozó is az adatok gyűjtésére, tárolására és az adatszolgáltatásra.

Ördögi körben élünk: nincs kereslet, ezért nincs adatbank, mert nincs adatbank, ezért a felhasználó nem keresi az adatokat.

Közben persze telik az idő, öregszenek a gépek.

KOVÁCS GYŐZŐ

„Memory" suli

Egy tantárgyfüggetlen oktatójáték

A program a hagyományos „memory" játék elvén működik, de nem az összetartozó képeket, hanem a fogalmakat kell megkeresni. A program a fogalompárok cseréjével szinte bármely tantárgyban alkalmas a tanultak játékos bevésésére és ellenőrzésére.

Ebben a verzióban egy történelem memoryt mutatunk be, ahol az egyes „kártyákon" évszámok és a hozzájuk tartozó név vagy esemény szerepel. 64 kártyával játszunk, tehát 32 párt kell megtalálni. A program futtatása során először a lerakott kártyákat soronként felemeljük, megjegyezzük. Ha az összes kártyával végeztünk, akkor kezdődik a tulajdonképpeni játék. A kártyákat egy felfelé nyíl (↑) mozgatásával tudjuk kiválasztani, majd a RETURN lenyomásával felfordítani. A nyilat a kurzorvezérlő gombokkal irányíthatjuk. Ha a játékot az összes pár megtalálása előtt be akarjuk fejezni, akkor a kurzorvezérlő gombok helyett a *-ot nyomjuk le.

A program BASIC-ben íródott, ráadásul kicsit szószátyár is, hogy a kevésbé gyakorlott programozók is bátran alakítsák. Különösebb magyarázatot nem igényel, esetleg a listán JÁTÉK címmel szereplő részt érdemes egy kicsit elemezni.

2310—2320 A nyíl kirajzolása az első kártya alá.
2410 A nyíl mozgatása.
2420—2430 A nyíl által kijelölt kártya felvétele.
2440—2480 A kártyán levő szöveg kiírása; a próbálkozások száma eggyel nő.
2500 A nyíl mozgatása.
2510—2520 A kiválasztott (második) kártya felfordítása.
2530—2570 Úgyanaz, mint a 2440—2480.
2580 A kártya indexének összehasonlítása.
2590—2670 A második kártya nem jó, visszafordítjuk.
2690—2820 A második kártya jó; a kártyát kivesszük, és a párok száma eggyel nő.

A program alakításával kapcsolatban a következőket ajánlom:

— Akinek van ékezetes gépe vagy betölthető betűkészlete, az ékezetes betűkkel dolgozzon.

— Célszerű az adatokat nem DATA sorokban, hanem lemezen vagy kazettán tárolni, és egy menüben onnan behívni (több tantárgynál akár mindegyiket külön lemezen, témakörönként).

— Ha a játék így túl nehéz (különösen általános iskolában), csökkenteni lehet a kártyák számát.

```
1000 REM **********************
1010 REM *                    *
1020 REM *     MEMORY SULI    *
1030 REM *                    *
1040 REM *  COMMODORE  PLUS/4  *
1050 REM *                    *
1060 REM * DEMETER LASZLO 1987 *
1070 REM *                    *
1080 REM **********************
1090 DIM C(64),V$(64),V(64)
1100 REM **********
1110 REM * CIMLAP *
1120 REM **********
1130 PRINT"■":PRINTCHR$(14)
1140 COLOR0,1,0:COLOR4,1,0
1150 PRINTTAB(9)"■■■M E M O R Y  S U L I"
1160 PRINTTAB(9)"■■     OKTATOJATEK"
1170 PRINTTAB(8)"■■■(C) DEMETER LASZLO 1987"
1180 PRINTTAB(3)"■■■A HAGYOMANYOS MEMORY ELVEN MUKODO"
1190 PRINTTAB(3)"■OKTATOJATEK, EGYMASSAL OSSZEFUGGO"
1200 PRINTTAB(3)"■FOGALMAK BEVESESENEK MEGKONNYITE-"
1210 PRINTTAB(3)"■SERE  TETSZOLEGES  TANTARGYAKBAN."
1220 PRINTTAB(1)"■■■TURELEM! ■TÖRTENELEM■ADATOK BEOLVASASA.";
1230 REM ********************
1240 REM * CIMEK BEOLVASASA *
1250 REM ********************
1260 FOR J1=0 TO 840 STEP 120
1270 FOR J2=0 TO 21  STEP 3
1280 I=(J1/120)*8+(J2/3)+1
1290 C(I)=J1+J2
1300 NEXT
1310 NEXT
1320 REM ********************
1330 REM * SZAVAK BEOLVASASA *
1340 REM ********************
1350 FOR J1=0 TO 31
1360 FOR J2=1 TO 2
1370 READ V$(J1*2+J2)
1380 V(J1*2+J2)=J1+1
1390 NEXT
1400 NEXT
1410 REM **********************
1420 REM * A SZAVAK KEVERESE *
1430 REM **********************
1440 FOR JJ=1 TO 100
1450 R1=INT((RND(0)*64)+1)
1460 R2=INT((RND(0)*64)+1)
1470 SV$=V$(R1)
1480 SV=V(R1)
1490 V$(R1)=V$(R2)
1500 V(R1)=V(R2)
1510 V$(R2)=SV$
1520 V(R2)=SV
1530 NEXT
1540 REM ********************
1550 REM * A JATEKTER RAJZA *
1560 REM ********************
1570 PRINT"■":COLOR4,6,5
1580 FOR JJ=0 TO 23
1590 POKE 3096+40*JJ,160
1600 POKE 2072+40*JJ,85
1610 NEXT
1620 FOR JJ=0 TO 39
1630 POKE 4032+JJ,160
1640 POKE 3008+JJ,85
1650 NEXT
```

```
1660 REM ********************
1670 REM * A LAPOK LERAKASA *
1680 REM ********************
1690 PRINT"■"
1700 PRINTTAB(27)"■■ELOSZOR"
1710 PRINTTAB(27)"■RAKJUK LE"
1720 PRINTTAB(27)"■A LAPOKAT"
1730 PRINTTAB(27)"■HATUKKAL"
1740 PRINTTAB(27)"■FELFELE."
1750 FOR K=1 TO 3000:NEXT
1760 FOR JJ=1 TO 24
1770 POKE 3113+C(JJ),160
1780 POKE 2089+C(JJ),72
1790 NEXT
1800 REM ************************
1810 REM * A LAPOK FELFORDITASA *
1820 REM ************************
1830 PRINT"■"
1840 FOR JJ=1 TO 12
1850 PRINTTAB(26)"■■                    ■"
1860 NEXT
1870 PRINT"■"
1880 PRINTTAB(27)"■■EZUTAN"
1890 PRINTTAB(27)"■FIGYELMESEN"
1900 PRINTTAB(27)"■EMELJUK FEL"
1910 PRINTTAB(27)"■SORONKENT"
1920 PRINTTAB(27)"■A LAPOKAT."
1930 FOR K=1 TO 3000:NEXT
1940 PRINT"■"
1950 FOR JJ=1 TO 12
1960 PRINTTAB(26)"■■                    ■"
1970 NEXT
1980 FOR I1=0 TO 7
1990 PRINT"■";
2000 FOR I2=1 TO 8
2010 POKE 3113+C(I1*8+I2),160
2020 POKE 2089+C(I1*8+I2),113
2030 PRINTTAB(26)"■■";V$(I1*8+I2)
2040 FORK=1TO1000:NEXT
2050 NEXT
2060 PRINTTAB(26)"■■■TOVABB A"
2070 PRINTTAB(26)"■■RETURN■-NEL!"
2080 GETQ$:IFQ$<>CHR$(13)THEN2080
2090 PRINT"■"
2100 FOR JJ=1 TO 23
2110 PRINTTAB(26)"■■                    ■"
2120 NEXT
2130 FOR I2=1 TO 8
2140 POKE 3113+C(I1*8+I2),160
2150 POKE 2089+C(I1*8+I2),72
2160 NEXT
2170 NEXT
```

```
2180 REM ***********
2190 REM * A JATEK *
2200 REM ***********
2210 PRINT"■"
2220 PRINTTAB(27)"■A FIGYELEM!"
2230 PRINTTAB(27)"■■KEZDODHET"
2240 PRINTTAB(27)"■A JATEK!"
2250 FOR K=1 TO 3000:NEXT
2260 PRINT"■"
2270 FOR JJ=1 TO 12
2280 PRINTTAB(26)"■■                    ■"
2290 NEXT
2300 I=1:PR=0:PA=0
2310 POKE 3153+C(I),30
2320 POKE 2129+C(I),118
2330 PRINT"■"
2340 PRINTTAB(26)"■1. LAP:"
2350 PRINTTAB(26)"■■■■2. LAP:"
2360 PRINT"■■■■■■■■"
2370 PRINTTAB(26)"■PROBALKOZAS:"
2380 PRINTTAB(26)"■";PR
2390 PRINTTAB(26)"■■PAROK SZAMA:"
2400 PRINTTAB(26)"■";PA
2410 GOSUB 3010
2420 POKE 3113+C(I),160
2430 POKE 2089+C(I),0
2440 PRINT"■"
2450 PRINTTAB(26)"■■■";V$(I)
2460 PR=PR+1
2470 PRINT"■■■■■■■■■■■■■■■■■■■■"
2480 PRINTTAB(26)"■";PR
2490 W=I
2500 GOSUB 3010
2510 POKE 3113+C(I),160
```

```
2520 POKE 2089+C(I),113
2530 PRINT"■"
2540 PRINTTAB(26)"■■■■■■■■";V$(I)
2550 PR=PR+1
2560 PRINT"■■■■■■■■■■■■■■■■■■■■"
2570 PRINTTAB(26)"■";PR
2580 IF V(I)=V(W) THEN 2690
2590 PRINT"■"
2600 PRINTTAB(26)"■■■■■■■■■■NEM JO!"
2610 FOR K=1 TO 3000:NEXT
2620 PRINT"■■■■■■■■";
2630 FOR JJ=1 TO 6
2640 PRINTTAB(26)"■■                    ■"
2650 NEXT
2660 POKE 3113+C(I),160
2670 POKE 2089+C(I),72
2680 GOTO 2500
2690 PRINT"■"
2700 PRINTTAB(26)"■■■■■■■■■■SIKERULT!"
2710 PA=PA+1
2720 PRINT"■■■■■■■■■■■■■■■■■■■■"
2730 PRINTTAB(26)"■";PA
2740 FOR K=1 TO 3000:NEXT
2750 PRINT"■"
2760 PRINTTAB(26)"■■■■                  ■"
2770 PRINT"■■■■■■■■";
2780 FOR JJ=1 TO 6
2790 PRINTTAB(26)"■■                    ■"
2800 NEXT
2810 POKE 3113+C(I),160
2820 POKE 2089+C(I),0
2830 IF PA=32 THEN 2880
2840 GOTO 2410
```

— Átírható a program úgy is, hogy egyszerre ne csak egy játékos játszhassék, vagy úgy is, hogy egyszerű képek is megjeleníthetők legyenek.

Végül — elsősorban azokat segítve, akiknek nincs Commodore Plus/4-es gépük, de vállalkoznak a program átírására, — a táblázatban megadom a programlistában szereplő speciális karakterek jelentését is.

DEMETER LÁSZLÓ

A rovatvezető kiegészítése. A program nem jegyzi meg, hogy egy lap már fel van fordítva. Újra meg lehet fordítani, sőt ezt jónak fogadja el, ha a saját párjának tekintem.

Olvasóinktól *várjuk a program bővítését* e „csalás" lehetőségének megakadályozására!

```
2850 REM **************
2860 REM * BEFEJEZES *
2870 REM **************
2880 PRINT"■" COLOR4,1,0
2890 SZ=INT((2*PA/PR)*10000)/100
2900 PRINT"■■  A JATEK BEFEJEZODOTT."
2910 PRINT"■■  A PROGRAM UJRAINDITASA AZ"
2920 PRINT"■■  F6 ■■OMBBAL LEHETSEGES."
2930 PRINT"■■■  A VEGEREDMENY:"
2940 PRINT"■  A PROBALKOZASOK SZAMA  ";PR
2950 PRINT"■  A MEGTALALT PAROK SZAMA:";PA
2960 PRINT"■  AZ EREDMENY SZAZALEKBAN ";SZ
2970 PRINT"■" END
2980 :
2990 :
3000 :
3010 REM ****************************
3020 REM * CURSOR MOZGATO SZUBRUTIN *
3030 REM ****************************
3040 GETQ$:IFQ$=""THEN 3040
3050 IF Q$="■" THEN J=I-1:GOTO 3120
3060 IF Q$="■" THEN J=I+1:GOTO 3120
3070 IF Q$="■" THEN J=I-8:GOTO 3120
3080 IF Q$="■" THEN J=I+8:GOTO 3120
3090 IF Q$=CHR$(13) THEN 3190
3100 IF Q$="■" THEN 2880
3110 GOTO 3040
3120 IF J<1 OR J>64 THEN J=I
3130 POKE 3153+C(I),160
3140 POKE 2129+C(I),0
3150 I=J
3160 POKE 3153+C(I),30
3170 POKE 2129+C(I),118
3180 GOTO 3040
3190 RETURN
3200 :
3210 :
3220 :
```

```
3230 REM ************************
3240 REM * EVSZAMOK, FOGALMAK   *
3250 REM *   A KOZEPISKOLAS     *
3260 REM * TORTENELEM ANYAGBOL *
3270 REM ************************
3280 DATA"896.","HONFOGLALAS"
3290 DATA"955.","AUGSBURG"
3300 DATA"997-1038","SZENT ISTVAN"
3310 DATA"1077-1095","SZENT LASZLO"
3320 DATA"1222.","ARANYBULLA"
3330 DATA"1235-1270","IV. BELA"
3340 DATA"1241.APR.11.","MUHI CSATA"
3350 DATA"1308-1342","KAROLY ROBERT"
3360 DATA"1342-1382","I. NAGY LAJOS"
3370 DATA"1444.NOV.10.","VARNAI CSATA"
3380 DATA"1448.OKT.19.","RIGOMEZO"
3390 DATA"1458-1490","MATYAS"
3400 DATA"1485.","BECS ELFOGL."
3410 DATA"1514.","DOZSA"
3420 DATA"1526.AUG.29.","MOHACS"
3430 DATA"1538.","VARADI BEKE"
3440 DATA"1541.","BUDA ELFOGL."
3450 DATA"1566.","SZIGETVAR"
3460 DATA"1606.","BECSI BEKE"
3470 DATA"1703-1711","RAKOCZI FELK."
3480 DATA"1740-1780","MARIA TEREZIA"
3490 DATA"1780-1790","II. JOZSEF"
3500 DATA"1848.APR.11.","APR.TORVENYEK"
3510 DATA"1848.OKT.7.","OZORA"
3520 DATA"1849.APR.6.","ISASZEG"
3530 DATA"1849.AUG.13.","VILAGOS"
3540 DATA"1849.OKT.6.","ARAD"
3550 DATA"1867.","KIEGYEZES"
3560 DATA"1914-1918","I. VILAGHAB."
3570 DATA"1919.MARC.21.","TANACSKOZT."
3580 DATA"1939-1945","II. VILAGHAB."
3590 DATA"1945.APR.4.","FELSZABADULAS"
```

Gépi kódban

Box-utasítás PRIMO

A Primo felhasználhatóságát bővíthetjük egy olyan gépi kódban írt programrésszel, amely a és b oldalhosszúságú téglalapok és négyzetek kirajzolására alkalmas. A bal alsó csúcspont koordinátái az x és az y értékkel adhatók meg. Az a = 1, illetve b = 1 esetekben függőleges és vízszintes vonalakat is rajzolhatunk. Ezeket a G9 jelű programban *(1. lista)* próbálhatjuk ki. A BASIC rész átírásával különféle játékok helyszínét építhetjük fel (labirintus, várfal stb.).

A gépi kódú programrész két egymásba ágyazott ciklust tartalmaz, ezekben az előző cikkeimben szereplő utasításokon kívül egy újat, az ADD A, (IY+eltolás) utasítást is használjuk. Ez az A regiszterben levő értékhez hozzáadja az (IY+eltolás) címen levő értéket, és az eredményt az A regiszterbe írja.

A G9-cel látszólag azonos feladat elvégzésére alkalmas a G10 jelű program is *(2. lista)*, de a gépi kódú programrész az egyes pontok kirajzolása előtt megvizsgálja az adott pontot a képernyőn, és ha bekapcsolt, akkor kikapcsolja, ellenkező esetben pedig bekapcsolja. Ez jól látható a következő értékek megadása esetén: a = 180, b = 30, x = 1 és y = 1.

A BASIC rész átírásával a képernyő különböző részeit jelölhetjük meg, mert azonos paraméterekkel a gépi kódú részt ismételten meghíva, a „jelölés" nyomtalanul megszűnik. Ezt felhasználhatjuk olyan programoknál, ahol háttér előtt kell egy négyzetet vagy téglalapot mozgatni, illetve egy adott terület be- és kikapcsolt állapotát kell váltogatni (egy pont megjelölése a térképen, egy szövegrész kiemelése a képernyőn stb.).

Ebben a programban új utasítás a SUB D, amely az A regiszterben levő értékből levonja a D regiszterben levő értéket, és az eredményt az A regiszterben tárolja. E műveletre azért van szükség, mert a 137. címen kezdődő szubrutin a D regiszterben levő értéket megváltoztatja, és ezt a változást hatástalanítani kell a 131., illetve a 134. címen kezdődő szubrutinok meghívása előtt.

SOMOGYI GYÖRGY

1. lista

```
10 REM G9
100 C=0 :D=0 :E=0 :X=0 :Y=0 :A=0 :B=0
110 DIM K%(20),L%(20)
120 C=VARPTR(K%(0))
122 D=VARPTR(L%(0))
140 POKEC,213,253,225,253,70,1,253,112,
      1,253,70,0,120,253,134,2,87,253,
      126,1,253,134,3,95,205,131,0,16,
      239,253,70,1,16,228,201
150 CLS
157 INPUT"A=";A
159 INPUT"B=";B
161 INPUT"X=";X
163 INPUT"Y=";Y
170 POKED,A,B,X,Y
180 E=CALL(C,D)
```

A PROGRAM	A KOD
PUSH DE	213
POP IY	253,225
LD B,(IY+1)	253,70,1
LD(IY+1),B	253,112,1
LD B,(IY+0)	253,70,0
LD A,B	120
ADD A,(IY+2)	253,134,2
LD D,A	87
LD A,(IY+1)	253,126,1
ADD A,(IY+3)	253,134,3
LD E,A	95
CALL,131	205,131,0
DJNZ(-17)	16,239
LD B,(IY+1)	253,70,1
DJNZ(-28)	16,228
RET	201

2. lista

```
10 REM G10
100 C=0 :D=0 :E=0 :X=0 :Y=0 :A=0 :B=0
110 DIM K%(30),L%(20)
120 C=VARPTR(K%(0))
122 D=VARPTR(L%(0))
140 POKEC,213,253,225,253,70,1,253,112,
      1,253,70,0,120,253,134,2,87,253,
      126,1,253,134,3,95,14,10,205,137,
      0,40,2,14,11,62,191,146,87,121,
      254,11,40,5,205,131,0,24,3,205,
      134,0,16,216,253,70,1,16,205,201
150 CLS
157 INPUT"A=";A
159 INPUT"B=";B
161 INPUT"X=";X
163 INPUT"Y=";Y
170 POKED,A,B,X,Y
180 E=CALL(C,D)
190 FOR E=1 TO 500 : NEXT E
200 CLS
```

A PROGRAM	A KOD
PUSH DE	213
POP IY	253,225
LD B,(IY+1)	253,70,1
LD(IY+1),B	253,112,1
LD B,(IY+0)	253,70,0
LD A,B	120
ADD A,(IY+2)	253,134,2
LD D,A	87
LD A,(IY+1)	253,126,1
ADD A,(IY+3)	253,134,3
LD E,A	95
LD C,10	14,10
CALL,137	205,137,0
JR Z,2	40,2
LD C,11	14,11
LD A,191	62,191
SUB D	146
LD D,A	87
LD A,C	121
CP,11	254,11
JR Z,5	40,5
CALL,131	205,131,0
JR,3	24,3
CALL,134	205,134,0
DJNZ,(-40)	16,216
LD B,(IY+1)	253,70,1
DJNZ,(-51)	16,205
RET	201

DIÁK C64 ROVAT

Nyújtott karakterek

A program a kurzor aktuális pozíciójától kezdve, Y irányban kétszeresre nyújtva írja ki az A$ változóban tárolt szöveget. Csak a betűket nyújtja, a többi karakter helyére szóközt ír.

A BASIC programot *(1. lista)* beírása után mentsük ki. Futtatva előbb normál méretben, majd nyújtva írja ki a betűket. A 17—22-es sorok csak példát mutatnak a kiírásra.

A rutin két részből áll. Az első (100—150-es sorok) az inicializáló rész, a gépi kódot tölti be. Saját programjainkban csak egyszer, azok elején hívjuk meg a GOSUB100 utasítással.

1. lista

```
10 REM
11 REM  87(C) PANDA SOFT KUMMERT AKOS
12 REM
13 REM  VAGI E.SZ.I
14 REM
15 REM  NYUJTOTT KARAKTEREK
16 REM
17 GOSUB100:PRINT"▓▓▓▓▓▓          ";:A$=""
20 FORL=0TO26:A$=A$+CHR$(64+L):NEXT
21 PRINTA$:FORH=1TO10:PRINT"         ";
22 GOSUB200:NEXT:END
100 READX:IFX<0THEN120
110 POKE49152+A,X:A=A+1:S=S+X:GOTO100
120 IFS=16502THENSYS49152:RETURN
121 PRINT"▓▓HIBA A DATA SORBAN !":END
130 DATA 120,169,  0,133, 28,133, 30
131 DATA 169,208,133, 29,169,240,133
132 DATA  31,162, 16,160,  0,169, 51
133 DATA 133,  1,177, 28,230,  1,145
134 DATA  30,200,208,243,230, 29,230
135 DATA  31,202,208,234,169,  0,133
136 DATA  28,169,208,133, 29,169,  0
137 DATA 133, 30,169,242,133, 31,169
138 DATA  51,133,  1,160,  0,177, 28
139 DATA 230,  1,145, 30,200,145, 30
140 DATA 230, 28,208,  2,230, 29,165
141 DATA  30, 24,105,  2,133, 30,144
142 DATA   2,230, 31,165, 31,201,243
143 DATA 208,218,165, 30,201,176,208
144 DATA 212,169, 55,133,  1, 88,169
145 DATA   0,141,134,  2,169,204,141
146 DATA 136,  2,169, 60,141, 24,208
147 DATA 169,  0,141,  0,221,169, 14
148 DATA 141, 32,208,169,  1,141, 33
149 DATA 208,169,147, 32,210,255, 96
150 DATA  -1
160 REM NYUJTOTT KARAKTERRE ATALAKITO
161 REM RUTIN: SYS49152
162 REM
163 REM A$ - AZ ATALAKITANDO SZOVEG
164 REM
165 REM A1$ - AZ ATALAKITOTT SZOVEG
166 REM       ELSO SORA
167 REM
168 REM A2$ - AZ ATALAKITOTT SZOVEG
169 REM       MASODIK SORA
200 A1$="":A2$="":K=0
201 FORL=1TOLEN(A$):K=K+1
202 IFK=41THENK=-1:GOSUB209
203 X=(ASC(MID$(A$,L,1)))AND191
204 IFX<27THENGOSUB213
205 IFX<>26THENGOSUB215
206 NEXT
207 IFA1$<>""THENGOSUB209
208 RETURN
209 IFK=-1THENPRINTA1$;A2$:GOTO212
210 P=PEEK(211):PRINTA1$
211 POKE211,P:PRINTA2$
212 A1$="":A2$="":K=0:RETURN
213 A1$=A1$+CHR$((X*2)OR192)
214 A2$=A2$+CHR$((X*2+1)OR192):RETURN
215 A1$=A1$+" ":A2$=A2$+" ":RETURN
```

```
., C000  78        SEI          Átmásolja a ROM karaktereket a KERNALROM
., C001  A9 00     LDA #$00     alatt lévő RAM-ba.
., C003  85 1C     STA $1C
., C005  85 1E     STA $1E      $D000-$E000 ROM-ból a $E000-$FFFF RAM-ba.
., C007  A9 D0     LDA #$D0
., C009  85 1D     STA $1D
., C00B  A9 F0     LDA #$F0
., C00D  85 1F     STA $1F
., C00F  A2 10     LDX #$10     2K-t fog átmásolni
., C011  A0 00     LDY #$00
., C013  A9 33     LDA #$33     Karakter ROM bekapcsolva
., C015  85 01     STA $01
., C017  B1 1C     LDA ($1C),Y  A byte az akkumlátorban van
., C019  E6 01     INC $01      Karakter ROM vissza kapcsolása
., C01B  91 1E     STA ($1E),Y  A byte beírása a RAM-ba
., C01D  C8        INY
., C01E  D0 F3     BNE $C013
., C020  E6 1D     INC $1D      A következő 256 byte
., C022  E6 1F     INC $1F
., C024  CA        DEX
., C025  D0 EA     BNE $C011
., C027  A9 00     LDA #$00     A nyújtott karakterek átmásolása a ROM-ból.
., C029  85 1C     STA $1C
., C02B  A9 D0     LDA #$D0
., C02D  85 1D     STA $1D      $D000-tól
., C02F  A9 00     LDA #$00
., C031  85 1E     STA $1E      $F200-ba
., C033  A9 F2     LDA #$F2
., C035  85 1F     STA $1F
., C037  A9 33     LDA #$33     A karakter ROM bekapcsolása
., C039  85 01     STA $01
., C03B  A0 00     LDY #$00
., C03D  B1 1C     LDA ($1C),Y  A byte az akkumlátorban van
., C03F  E6 01     INC $01      Karakter ROM vissza kapcsolása
., C041  91 1E     STA ($1E),Y  A byte beírása a RAM-ba (először)
., C043  C8        INY          Cím növelése
., C044  91 1E     STA ($1E),Y  A byte beírása a RAM-ba (másodszor)
., C046  E6 1C     INC $1C      A ROM-mutató növelése eggyel
., C048  D0 02     BNE $C04C
., C04A  E6 1D     INC $1D
., C04C  A5 1E     LDA $1E      A RAM-mutató növelése kettővel
., C04E  18        CLC
., C04F  69 02     ADC #$02
., C051  85 1E     STA $1E
., C053  90 02     BCC $C057
., C055  E6 1F     INC $1F
., C057  A5 1F     LDA $1F      Ha a RAM-mutató $F3B0 akkor vége
., C059  C9 F3     CMP #$F3
., C05B  D0 DA     BNE $C037
., C05D  A5 1E     LDA $1E
., C05F  C9 B0     CMP #$B0
., C061  D0 D4     BNE $C037
., C063  A9 37     LDA #$37     Vissza kapcsolás az eredeti ROM-ra
., C065  85 01     STA $01
., C067  58        CLI
., C068  A9 00     LDA #$00     A karakterszín beállítása (FEKETE)
., C06A  8D 86 02  STA $0286
., C06D  A9 CC     LDA #$CC     A képernyő átrakása a $CC00 területre
., C06F  8D 88 02  STA $0288
., C072  A9 3C     LDA #$3C     A karakterkészlet átállítása $E000 címre
., C074  8D 18 D0  STA $D018
., C077  A9 00     LDA #$00     Az utolsó 16K-s szelet kiválasztása
., C079  8D 00 DD  STA $DD00
., C07C  A9 0E     LDA #$0E     A képernyő keret színe (VILÁGOSKÉK)
., C07E  8D 20 D0  STA $D020
., C081  A9 01     LDA #$01     A képernyő háttér színe (FEHÉR)
., C083  8D 21 D0  STA $D021
., C086  A9 93     LDA #$93     Képernyő törlése
., C088  20 D2 FF  JSR $FFD2
., C08B  60        RTS
```

2. lista

A második rész (200—215-ös sorok) maga a nyújtott karaktereket kiíró rutin. Hívása előtt tegyük A$-ba a kiírandó szöveget, majd közönséges PRINT utasítással pozícionáljuk a kurzort a képernyő kívánt helyére, végül a GOSUB200 utasítással hívjuk meg a rutint. Ha a kiírandó szöveg nem fér el a kezdő sorban, akkor egy sor kihagyása után a következő sor elején folytatódik.

Azok számára, akik mélyebben meg akarják érteni az eljárás működését, közlöm a gépi kód magyarázatokkal kiegészített assembler programját *(2. lista).*

KUMMERT ÁKOS

Finom kiíratás Primóra

Egy Primo B—32 számítógép tulajdonosa vagyok egy éve. Még csak BASIC-ben programozok. Két észrevételemet közlöm, melyeknek bizonyára sok Primo-tulajdonos örül majd:

POKE 16456,X ahol 0< =X< =255
POKE 16457,Y ahol 0< =Y< =179

Ezzel az utasításpárral raszterpontnyi pontossággal helyezhetünk el karaktereket a képernyőre. Az első utasítás a vízszintes, a második a függőleges elhelyezését adja meg.

Bemutatok egy rövid példaprogramot:

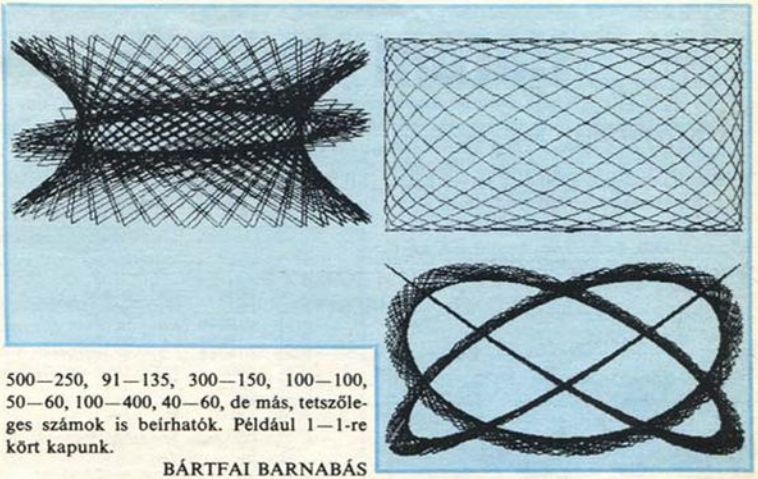
```
10 CLS
20 POKE 16456,98:POKE 16457,86
30 PRINT
CHR$(2)"PRIMO"CHR$(1)
40 END
```

A CHR$(2) bekapcsolja, a CHR$(1) pedig kikapcsolja a dupla szélességű írást.

KOVÁCS TAMÁS

DIÁK C64 ROVAT

Grafika



Az alábbi programok igen változatos ábrákat csalnak a képernyőre.

Az első *(1. lista)* Simon's BASIC-ben készült, így a C16-on való futtatáshoz csupán a 95-ös, 130-as és 160-as sorokat kell átírni, a 170-es, 180-as, 190-es sorokat pedig kihagyni.

Ha netán nem áll rendelkezésünkre C64-en Simon's BASIC, akkor a *2. listán* közölt programot gépeljük be. A futási idő ugyan lényegesen megnő, viszont elleshetjük belőle a grafika CBM BASIC-ben történő programozását. Futtatáskor a „szorzók" kérdésre a következő számpárokat ajánlom:90—135, 500—400, 12—13, 500—250, 91—135, 300—150, 100—100, 50—60, 100—400, 40—60, de más, tetszőleges számok is beírhatók. Például 1—1-re kört kapunk.

BÁRTFAI BARNABÁS

```
10 PRINT"■■■■■■■■ATOMSOFT B.B."
20 PRINT"■ HA UGY GONDOLJA , HOGY ELEG"
25 PRINT" NYOMJON MEG EGY GOMBOT !"
30 PRINT"■ AJANLOM A KOVETKEZO SZAMOKAT:"
40 PRINT"■ 90-135;500-400;12-13;500-250;91-135"
50 PRINT"■ 300-150;100-100;50-60;40-60;100-400"
80 PRINT
90 INPUT SX,SY
95 HIRES 1,2:W=160:V=190
100 I=I+.02
110 X=SIN(I*SX)*150+160
120 Y=COS(I*SY)*90 +100
130 LINE W,V,X,Y,1
140 W=X:V=Y
150 GETT$:IF T$="" THEN 100
160 CSET 0
170 INPUT"PLOTTERRE KIRAJZOLJAM  /I,N/";P$
180 IF P$<>"I" AND P$<>"Y" THEN RUN
190 COPY
200 RUN
```

1. lista

2. lista

```
 0 POKE 54296,15:X1=160:Y1=190
 2 PRINT"■■■       ATOMSOFT 1986"
 5 INPUT"■■■SZORZOK";SA,SB
 10 BA=8192:POKE 53272,PEEK(53272)OR8
 30 POKE 53265,PEEK(53265)OR 32
 50 FOR I=BA TO BA+7999 :POKE I,0:NEXT
 70 FOR I=1024 TO 2023 :POKE I,1:NEXT
 100 FOR P=0 TO 7.6 STEP .02
 110 X2=INT(160+SIN(P*SA)*150)
 120 Y2=INT(100+COS(P*SB)*90)
 130 GOSUB 500
 140 X1=X2:Y1=Y2
 150 NEXT P
 155 POKE 53265,27:POKE 53272,21:END
 160 CH=INT(X/8):RO=INT(Y/8)
 170 LN=Y AND 7
 180 BY=BA+RO*320+8*CH+LN
 190 BI=7-(X AND 7)
 200 POKE BY,PEEK(BY)OR(2↑BI)
 210 RETURN
 220 POKE 53265,27:POKE 53272,21
 500 KX=X1-X2:KY=Y1-Y2
 501 KB=1:KC=1
 502 IF X1>X2 THEN KB=-1
 503 IF Y1>Y2 THEN KC=-1
 510 IF KY=0 THEN 3000
 520 IF KX=0 THEN 4000
 530 IF ABS(KX)>ABS(KY) THEN 1000
 540 IF ABS(KX)<ABS(KY) THEN 2000
 550 FOR I=X1 TO X2 STEP KB
 560 X=I:Y=Y1+I-X1
 570 GOSUB 160
 580 RETURN
 1000 KH=KY/ABS(KX):KO=Y1
 1010 FOR I=X1 TO X2 STEP KB
 1020 KO=KO-KH:Y=KO:X=I
 1030 GOSUB 160
 1035 NEXT
 1040 RETURN
 2000 KH=KX/ABS(KY):KO=X1
 2010 FOR I=Y1 TO Y2 STEP KC
 2020 KO=KO-KH:X=KO:Y=I
 2030 GOSUB 160
 2035 NEXT
 2040 RETURN
 3000 FOR I=X1 TO X2 STEP KB
 3010 Y=Y1:X=I
 3020 GOSUB 160
 3025 NEXT
 3030 RETURN
 4000 FOR I=Y1 TO Y2 STEP KC
 4010 X=X1:Y=I
 4020 GOSUB 160
 4025 NEXT
 4030 RETURN
```

DIÁK spectrum ROVAT

Három rutin

A PATTERN rutin *(1. lista)* a képernyőre rajzolást könnyíti meg azzal, hogy egy 8×8-as hálót rajzol a karakterhelyek határaira; így könnyebben eligazodhatunk a készülő rajzon. A közölt assembler lista kezdőcíme 40000, de bárhova betölthető, mert csak relatív ugrásokat tartalmaz. A programot egy assemblerrel írjuk be. Kimenteni a SAVE "név" CODE cím,28 paranccsal, aktivizálni a RANDOMIZE USR cím utasítással lehet.

```
AFC8  210558   LD   HL,#5800
AFCB  0618     LD   B,#18
AFCD  3638     LD   (HL),#38
AFCF  78       LD   A,B
AFD0  E601     AND  #01
AFD2  0F       RRCA
AFD3  0F       RRCA
AFD4  AE       XOR  (HL)
AFD5  77       LD   (HL),A
AFD6  C5       PUSH BC
AFD7  0620     LD   B,#20
AFD9  7E       LD   A,(HL)
AFDA  23       INC  HL
AFDB  EE40     XOR  #40
AFDD  77       LD   (HL),A
AFDE  10F9     DJNZ #AFD9
AFE0  C1       POP  BC
AFE1  10EA     DJNZ #AFCD
AFE3  C9       RET
```

1. lista

A TURN rutin *(2. lista)* a képernyő tartalmát cseréli fel egy, a memória 32768-as címétől kezdődő képpel, bitenkénti átforgatással, nyolc lépésben. A memóriában lévő kép kezdőcímét a DE regiszterpár tartalmazza, átírásával a kép kezdete megváltoztatható. A rutin a színeket, attributumokat nem kezeli. Bárhova tölthető, mert ez is csak relatív ugrásokat tartalmaz.

Mentése: SAVE "név" CODE cím,33

Aktivizálása: RANDOMIZE USR cím.

Az IM2 rutin *(3. lista)* a megszakítást használja programvédelem céljából. BASIC-betöltőjét és hívását közlöm. Saját programokba könnyen beépíthető. Hívása után BREAK-re a futó program RANDOMIZE USR 0-t hajt végre. Ha valaki nem akar ennyire szigorú lenni, akkor a lista 80-as sorának első két nulláját átírva (elől az alacsony, hátul a magas helyiértékű bit) saját gépi kódú rutin kezdetére ugorhat.

Mentése: SAVE "név" CODE 65279,30. Hívását a lista 50-es sora mutatja.

TÖRKÖLY LÁSZLÓ

```
9C40  0608     LD   B,#08
9C42  C5       PUSH BC
9C43  210040   LD   HL,#4000
9C46  110080   LD   DE,#8000
9C49  010018   LD   BC,#1800
9C4C  CB06     RLC  (HL)
9C4E  CB0E     RRC  (HL)
9C50  EB       EX   DE,HL
9C51  CB16     RL   (HL)
9C53  EB       EX   DE,HL
9C54  CB16     RL   (HL)
9C56  23       INC  HL
9C57  13       INC  DE
9C58  0B       DEC  BC
9C59  78       LD   A,B
9C5A  B1       OR   C
9C5B  20EF     JR   NZ,#9C4C
9C5D  C1       POP  BC
9C5E  10E2     DJNZ #9C42
9C60  C9       RET
```

2. lista

```
 5 CLEAR 65278
10 FOR a=65279 TO 65308
20 READ b
30 POKE a,b
40 NEXT a
50 RANDOMIZE USR 65300
60 DATA   1,255,254,205, 84, 31, 48
70 DATA   5,241,195, 56,  0,201,241
80 DATA 195,  0,  0,201,  0,  0,  0
90 DATA 243, 62,254,237, 71,237, 94
100 DATA 251,201
```

3. lista

ADOK—VESZEK —CSERÉLEK

Ebben a rovatban rövid, szöveges, a mikroszámítógépekkel kapcsolatos hirdetéseket közlünk. A díjszabás: közleteknek gépelt soronként (60 karakter) 100,- Ft, magánszemélyeknek az első sor 50,- Ft, minden további sor 20,-Ft. Az NJSZT tagjainak az első három sor ingyenes. Hirdetéseiket a szerkesztőség címére várjuk.

ADOK

ATARI 800XL számítógép DATA séttel áron alul eladó. Játékprogram is van. Debrecen /52/13-292 telefonon.

C64-es programokat adok-veszek-cserélek. A választokat listával kérem: Magyar Attila, Kapuvár, Lenin u. 10. 9330.

ZX81-es géphez 64 KB-os bővítő olcsón eladó. Levélcím: Vágner Gyula, Budapest, Hidegkúti út 80/a. 1028.

ZX-Spectrum programozható joystick interfésszel, kétezer programmal és igen bő szakirodalommal eladó 18 000,- Ft-ért. Cím: Magi István, Miskolc, Gyula út 54. 3/3. 3532.

Programok C/Plus 4 bővített **C16 számítógépekre**: FORTH nyelv /nem tiny-forth!/ 450 szó, 5 screen, diszk, printer és kiemelt magnokezelő utasítások. C/Plus 4 és C64 összehasonlító rutin- és rendszervátozó táblázat. Turbo programok /C64 Turbo-tape kompatibilis változat is!/. Turbo file: kazettás fájlkezelést tízszerre gyorsító program; BASIC bővítéssel. SYSTEM C/Plus 4 11 kbájtos rendszerprogram: BASIC és assembler fejlesztéshez. HEADER JUSTAGE: fejbeállító program. Disk-monitor. Turbocopy diszkhez és magnóhoz. C64-es program: MONITOR 'PLUS /a háttér RAM-okat is kezeli/. A programok ára: 180-600 Ft, kazettán, utánvéttel, részletes kezelési utasítással. Kérésre ismertetőt küldök. Cím: Pelsőczi Gyula, Szilasliget, Ady Endre út 36. 2145.

VESZEK

ATARI 800XL-hez magnót és CARTRIDGE-t veszek. Címem: Tudván /Solt, Pécs, Bárány Ignác u. 21. 7625.

C/Plus 4-es géphez olyan programokat veszek, ami feladni tudja a C64-es programokat. Cím: ifj. Futerán Barna, Encs, Borzsöny u. 24/a. 3860.

ZX-Spectrum nemképtelen gépeket veszek. Kemeri Károly, Kecskemét, Botond u. 2/C. IV/18. 6000.

11-99/AA számítógéphez 11 EXTENDED BASIC modult és hozzá tartozó leírást veszek. Egyéb programokat is érdekel. Kitka István, Szeged, Budapesti krt. 16/C. IV.9. 6723.

CSERÉLEK

C16-ra játékprogramokat cserélek. Gábor Szilárd, Kapuvár, Kossuth L. u. 2. IV/41. 9330.

Commodore 16 és Plus 4-es játékprogramokat cserélnék. A programlistát a következő címre kérem: I. Nagy József, Kecskemét, Fehér u. 20. 6000.

Commodore 16-os számítógépre játékprogramokat cserélnék. Válaszokat a programok listájával kérek: Kiss Gábor, Veszprém, Gábor Á. u. 2/C. I/4. 8200.

C64-re játék és egyéb programokat cserélek. Csak ismerős! Válaszokat listával kérem. Mityási Arnold, Jánoshalma, Kossuth u. 38. 6440. Tel.: 195.

Commodore 64-re készült játék és egyéb programokat cserélek. A válaszokat listával várom. Cím: Ajtai Zsolt, Monor, Dózsa György u. 35. 2200. Tel.: 363.

C64-re készült játékprogramot cserélek kazettán! Listát kérek: Imgrund Gábor, Bácsalmás, Árpád vezér u. 61. 6430.

Primo tulajdonosokkal programokat cserélek. Horváth László, 583-544 délelőtt.

TV COMPUTER-re veszek és cserélek játékprogramot. Viczián Norbert, Putnok, Fancsal u. 6. 3630.

VC20-as személyi számítógépemet és a Mikrovilág újságban eddig megjelent VC20 játékokat 8 k-s ZX81-es számítógépre cserélném. Ecsedi Gábor, Pécel, Pesti út 74. 2119.

ZX-Spectrumra néhány száz darabból álló programgyűjteményemet cserével szeretném kibővíteni. Választokat listával kérem. Szél János, Debrecen, Negyvötös u. 19. 4031.

OKTA-TOTÓ

A második forduló témaköre a hardver volt. Tekintve, hogy ez igen széles terület, a kérdések is — a nehézségi fokokkal együtt — meglehetősen változatosak voltak. Lássuk a helyes válaszokat.

A dobrendszerű sornyomtató egy nyomtatási ciklusa a dob akkora elfordulása, amely alatt a teljes karakterkészlet egyszer elhalad a kalapácssor előtt. Ezalatt egy teljes sort nyomtat ki (1).

Mágneslemezre soros adatállományok felírása szektoronként sorban történik. A fejmozgások optimalizálása érdekében cilinderenkénti felírást hajt végre (X).

Mágnesszalagokon szalagkezdetjel mindig van (néha kazettás szalagokon is), állománykatalógus és tisztító-befűző szalagdarab is lehet (1).

A központi egységnek természetesen az aritmetikai és vezérlőegység is része (X).

Az elektronikus operatív tárak jóval gyorsabbak a ma már elavultnak számító ferittáraknál, de hálózatkimaradás esetén elveszítik tartalmukat (a harmadik lehetőség egyszerűen értelmetlen volt) (2).

A vezérlőegység utasításregisztere — definíció szerint — az éppen végrehajtás alatt álló utasítást tartalmazza (2).

A mikroutasítás egy mikroprogram egyetlen utasítása. A mikroprocesszorok utasításai ugyanolyanok, mint bármely más processzor utasításai. A gépi utasítások részei pedig a műveleti kód és a címrész (1).

A C64 manapság körülbelül ugyanannyiba kerül, mint egy színes tv (2).

Minden számítógéprendszerre körülbelül egyformán jellemző, hogy a perifériák nem egymással, hanem külön-külön, a központi egységgel vannak összekötve (2).

A megszakításregiszter általában a megszakításkérésekről tartalmaz információt. A megszakítást okozó, illetve a megszakított eredményekről más egységek tárolnak adatokat (X).

A prioritás szó értelmezése szerint elsőbbséget jelent (2).

A perifériáknak sem a fizikai, sem a logikai egységszáma nem egyezhet meg, mivel ellenkező esetben vagy a hardver, vagy a programozó nem tudná őket megkülönböztetni (X).

A címvonalak az esetek 99 százalékában egyirányúak (kimenet) (1).

Az egycímes utasításnál a művelet rendszerint két operandussal zajlik. A második (a processzor által ismert) alapértelmezés (2).

A helyes megfejtés tehát összefoglalva:

1	2	3	4	5	6	7	8	9	10	11	12	13	+1
1	X	1	X	2	2	1	2	2	X	2	X	1	2

---

1. Mi küld információkat a felhasználói programok szintaktikus hibáiról?
1. az operációs rendszer
2. a fordítóprogramok
X. a szerkesztőprogramok

2. Egy program a futása során
1. sorban végrehajtja a benne foglalt utasításokat
2. egyszerre hajtja végre az összes utasítást
X. egyszerre hajt végre utasításcsoportokat

3. Egy beviteli-kiviteli utasítás végrehajtási sebessége alapvetően a
1. processzor
2. a használatos periféria
X. a felhasználói program sebességétől függ.

4. Általában mit értünk DOS alatt?
1. egy lemezen lévő felhasználói programok összességét
2. lemezkatalógust
X. lemezen lévő operációs rendszert

5. A BASIC interpreter
1. utasításonként vizsgálja a beírandó programot
2. soronként vizsgálja a beírandó programot
X. karakterenként vizsgálja az utasításokat

6. A FORTRAN programnyelv elsősorban
1. adatfeldolgozó
2. adatbázis-kezelő
X. tudományos-műszaki feladatok megoldására alkalmas.

7. Minél magasabb szintű programnyelven írjuk a felhasználói programokat
1. a kész program futása annál gyorsabb lesz
2. a programkészítés és tesztelés annál könnyebb lesz
X. a tárkihasználás annál jobb lesz.

8. A vezérlésátadó utasítások
1. megváltoztatják az utasításvégrehajtási sorrendet
2. átadják a vezérlést egy másik programnak
X. átadják a vezérlést az operációs rendszernek

9. Miért szükségesek a deklarációk?
1. kötelező szintaktikus előírások
2. fizikai tárterület lefoglalása a változóknak
X. enélkül nem futhatnak a programok

10. Ha egy program futása adathiba miatt megszakad, akkor
1. módosítjuk a bemenő adatokat
2. módosítjuk a programot
X. az elejétől - változatlanul - újraindítjuk a programot

11. Egy optimális BASIC program rendszerint
1. a lehető legkevesebb sorban elfér
2. a lehető legkevesebb DO utasítást tartalmazza
X. a lehető legkevesebb GOTO utasítást tartalmazza

12. A BASIC-ben a RETURN
1. a többi utasítástól függetlenül használható, önálló utasítás
2. nem használható
X. csak a GOSUB utasítással együtt használható

13. Egy programot hordozhatónak nevezünk, ha
1. különösebb módosítás nélkül többféle számítógéprendszeren lefuttatható
2. többféle fordítóprogrammal le lehet fordítani
X. működése független a bemenő adatoktól

13+1. A szerkesztő-összefűző programok
1. programjavítást tesznek lehetővé
2. különböző programmodulokat kapcsolnak össze egyetlen futtatható programmá
X. forrásnyelvű programrészleteket kapcsolnak össze

Beküldési határidő: július 20.

PÁLYÁZATI SZELVÉNY 87/7

OKTA-TOTÓ

Kérdés	Tipp
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	
11.	
12.	
13.	
+	
13+1.	

Név: _____

Cím: _____

_____

Szem.szám.

ATARI BASIC és lemezkezelés

Előző cikkünket azzal zártuk, hogy táblázatosan összevetettük a BASIC MS standard, C64 és A—800XL megfelelőit, megjegyezve, hogy a formai azonosság takarhat tartalmi különbözőségeket.

Ezt a részt a BASIC néhány működési sajátosságának leírásával kezdjük. A tárgyalást az egyes BASIC kulcsszavakhoz kötjük. A cikk második felében a lemezszervezéssel foglalkozunk.

BASIC

Egy olyan gépen, ahol nincs beépített PI konstans, különös jelentősége van az **ATN** utasításnak, melynek segítségével egyszerűen elő tudjuk állítani PI értékét:

```
A=4*ATN (1)
PRINT A
```

A értéke: 3,14159267

A **CLR**-t a BASIC egyik legkritikusabb utasításának tartjuk. Feladata az MS standardban — és tudomásunk szerint valamennyi általánosan használt BASIC implementációban — az, hogy törli a változóterületet, azaz megszünteti a területkijelölések (DIM) érvényét, és a felvett értékeket Ø-ra állítja.

Itt ez az utasítás sajátosan működik. Nem törli ténylegesen a változóterületet, bár újra dimenzionálhatóvá teszi. Mindössze egy mutató értékét állítja át, amely a DIM-ben kijelölt területek foglaltságára utal, illetve ezek szabad felhasználását engedélyezi.

Ennek az a következménye, hogy a tömbök újrafelhasználása esetén az előző felhasználás felül nem írt maradványát („szemetjét") megtaláljuk a tömbelemek értékében.

Nézzünk egy példaprogramot *(1. lista)*. A program alkalmas arra, hogy áthidaló, kényszermegoldásként egy karakteres tömböt kezeljen. A tömbelemeket a 30-as sorban fájlként megnyitott billentyűzetről olvassuk be karakterenként (60-80-as sor). A tömb öt, egyenként hétbetűs szót foglal magában. Ha a szó nem éri el a hétbetűs hosszt, RETURN-nel zárva a következő szó írása kezdődik. A program a szavak betűinek ATASCI kódját egy A(I,J) tömbbe teszi.

Indítsuk el újra meg újra a programot RUN-nal. Írjunk különböző hosszú szavakat az egyes sorokba. Figyeljük meg, hogy a régi szótöredékek a tömbben maradnak. Sajnos ezt csak a tömb felhasználása előtt, A(I,J)=Ø értékadással lehet kiküszöbölni.

A **LIST** utasítás segítségével nemcsak programot listázhatunk, hanem ún. listafájlokat is tudunk írni:

```
LIST"D: programnév. LST"
```

Ezeket csak az ENTER"D: programnév. LST" paranccsal kaphatjuk vissza. Ezek tulajdonképpen szövegfájlok, amelyeket például szövegszerkesztőbe is be tudunk olvasni.

Az **LPRINT** utasítás egyes esetekben (ha nem az 1029 típusú nyomtatót használjuk) nem működik. Így az EPSON FX 85, 105, 1000 típusú nyomtatóknál sem. Helyette a PRINT# utasítással jutunk azonos eredményhez.

A C64 BASIC alkalmazásainál sokszor gondot okozott a feltételes ciklusok szervezési lehetőségének hiánya. Jóllehet az MS BASIC tartalmazza a WHILE—WEND utasításpárt, ezt sem a C64, sem az A—800XL BASIC implementáció nem vette át.

Egyetlen ciklusszervezési lehetőség van: a FOR—NEXT utasításpárral. Ha azonban egy ilyen ciklus nem szabályosan, NEXT-en zárul — mivel mondjuk egy feltétel előbb teljesül, mintsem a ciklusváltozó felvenné a maximális értékét —, a ciklusból ki kell lépni. Új ciklus szervezése ilyenkor — különösen ugyanazzal a ciklusváltozóval — hibaüzenetet eredményezhet, és a program leállását, anélkül, hogy valójában hiba történt volna. Hasonló a helyzet akkor is, ha egy szubrutinba GOSUB-bal beléptünk a RETURN előtt — feltétel teljesülése miatt — lépünk ki.

Ennek a jelenségnek az oka a zsáktároló működése. A zsáktároló (verem) egy LIFO (Last In — First Out = utolsónak be, elsőnek ki) típusú átmeneti tároló: a ciklusok, szubrutinok visszatérési információinak ideiglenes tárolására szolgál. Így például a ciklusváltozó értéke és a GOSUB utáni visszatérési cím van a verem tetején az utoljára végrehajtott (FOR, GOSUB) utasításnak megfelelően. Ezt a NEXT, illetve a RETURN távolítja el.

Az A—800XL azonban gondoskodik egy utasításról, amely a ciklusokból, szubrutinokból való feltételes kilépést baj nélkül lehetővé teszi. Ez a **POP** utasítás. Akkor indokolt tehát használni, ha a ciklusokból való kilépés egy, a ciklusváltozó értékétől független feltétel teljesülése miatt válik szükségessé, illetve szubrutinból nem a RETURN-nél lépünk ki, vissza a GOSUB-ot követő utasításra, hanem például a szubrutinban kiszámított eredménytől függő pontjára a programnak.

Futtassuk le a következő példaprogramot *(2. lista)*. Ciklushibára utaló hibaüzenetet kapunk. Átalakítva a 12Ø-as sort:

```
12Ø IF J=12 THEN POP:GOTO 7Ø
```

a program hibaüzenet nélkül lefut.

A **POSITION** utasítás megfelel az MS BASIC LOCATE utasításának. Hatása ugyanaz, mint a C64-nél a

```
POKE 214,sor:POKE 211,oszlop:
SYS 58640:PRINT"szöveg"
```

parancssornak. Itt a formája:

```
POSITION oszlop,sor:PRINT"szöveg"
```

Megjegyezzük, hogy az ATARI LOCATE utasítása nem azonos ezzel.

1. lista

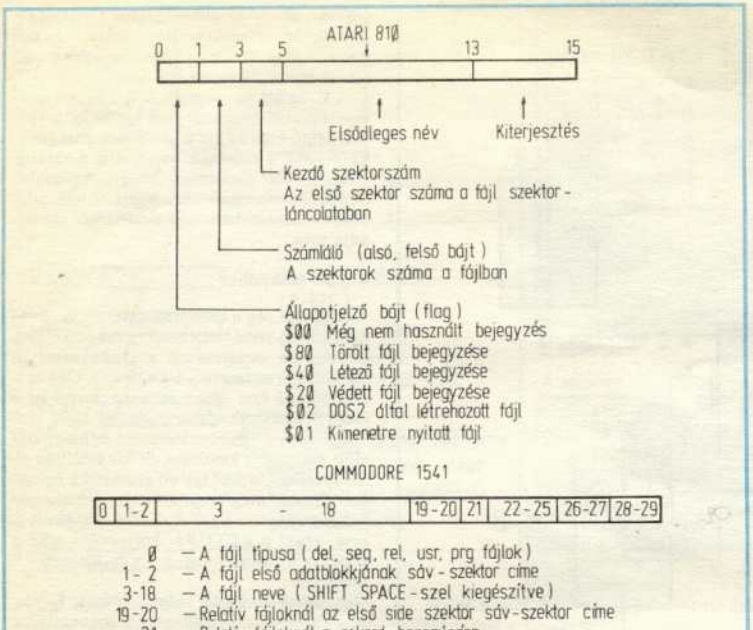
```
10 CLR
20 DIM A(5,7)
30 OPEN #1,4,0,"K:"
40 FOR I=1 TO 5
50 FOR J=1 TO 7
60 GET #1,B
70 IF B=155 THEN 110
80 A(I,J)=B
90 PRINT B;
100 NEXT J
110 PRINT
130 NEXT I
140 PRINT:PRINT
150 FOR I=1 TO 5
160 FOR J=1 TO 7
170 PRINT CHR$(A(I,J));
180 NEXT J
190 PRINT
200 NEXT I
```

2. lista

```
10 DIM A(20,20)
20 FOR I=1 TO 5
30 FOR J=1 TO 20
40 A(I,J)=I*J
50 GOSUB 120
60 NEXT J
70 FOR J=1 TO 20
80 PRINT J
90 NEXT J
100 NEXT I
110 END
120 IF J=12 THEN 70
130 PRINT J
140 RETURN
```

3. lista

```
10 DIM A$(20), B$(20)
20 A$="KARAKTERLANC"
30 B$="FUGGVENY"
40 PRINT A$(5)
50 PRINT A$(4,8)
60 A$(LEN(A$)+1)=B$
70 PRINT A$
```



1. ábra. Könyvtári bejegyzések

Sajátosan működik a **RUN** utasítás is. Ugyanis nem ad klirt (CLR-t), azaz nem törli a változóterületet. Ennek az a magyarázata, hogy a RUN utasítással egyúttal az MS standard CHAIN utasítását is meg tudjuk valósítani. Ennek az utasításnak az a szerepe, hogy egy behívott programnak átadja a hívó programban kiszámított, illetve ott használt változók értékeit. A hívó program azonban törlődik. Formája esetünkben:

RUN"D:programnév.BAS"

A karakterlánc-függvények használata is speciális. Míg a C64-nél a standardnak megfelelő LEFT$, RIGHT$, MID$ függvények segítségével volt megadható a részkarakterláncok kiemelése, levágása, addig az A—800XL esetében erre egyetlen forma szolgál. A karakterlánc-változó neve mellett zárójelben meg kell adni, hogy a kiemelés a lánc hányadik karakterénél kezdődik és hányadiknál végződik [például A$ (tólig)].

Tehát az A$(5) forma az A$ karakterláncot az 5. karakterétől a szó végéig kiírja. Az A$(4,8) a karakterláncot a 4. karakterétől a 8. karakterig adja vissza. Az összekapcsolás is speciális; példa rá a *3. lista.*

Az eredmény: AKTER
KTERLANC
KARAKTERLANCFUGG-
VENY

A karakterlánc-változók hosszát mindig dimenzionálni kell. A karakteres tömbkezelési műveletekre a következő cikkben térünk ki.

Lemezszervezés

Az FMS (File Management System) — ahogy már említettük — vezérli a fájlkezelést, szervezi a 720 szektor felhasználását a könyvtár, a lemez foglaltságát nyilvántartó táblázat (VTOC) és az adatszektorok segítségével.

A könyvtár

A könyvtár a $169-es (361.) szektorból indul, és folytatólagosan a $170. (368.) szektorig nyolc szomszédos szektort kapcsol össze. Minden szektor nyolc fájlbejegyzést tartalmazhat, ami lehetővé teszi, hogy egy lemezre összesen 64 (8 × 8) fájlt vigyünk fel.

Egy bejegyzés 16 bájt méretű, jellemző információt ad a feldolgozás alatt álló fájlok állapotáról, a fájlok szektorainak számáról. Tartalmazza továbbá a fájl nevét, az első szektor számát (amelyen az összefüggő szektorok láncolása indul), valamint a fájlkiterjesztést. Az *1. ábrán* összehasonlítás céljából ábrázoljuk az ATARI és a Commodore meghajtók könyvtári bejegyzéseit. Az ATARI 16 bájt hosszú bejegyzése mellett a C—1541-es meghajtónál ugyanez 30 bájt hosszúságú, részletezése az ábra szerinti. A könyvtárhoz kapcsolódó jellemző adatokat az *1. táblázattal* szemléltetjük. Végül a *2. ábrán* a nyolc könyvtárszektort és a hozzájuk kapcsolódó fájlbejegyzést mutatjuk be.

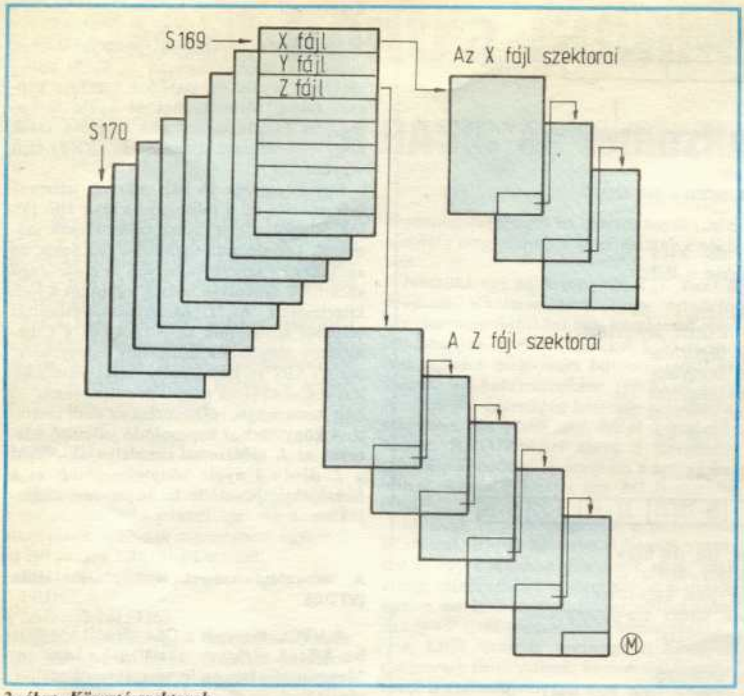
A lemezfoglaltságot nyilvántartó tábla (VTOC)

A VTOC szerepét a C64-nél a BAM tölti be. A $168. szektorban található a lemezen. Megmutatja, hogy a lemezszektorok melyik sávját foglalja el az adatfájl. A szektor szervezését a *3. ábra* szemlélteti.

A VTOC legfontosabb része a bittérkép. Ez egy folytatólagos, 90 bájt hosszú sztring, amelyben minden bájt 8 bitet tartalmaz. Így a bittérképen 720 (90 × 8) bit található. Ez pontosan annyi, mint a szektorok száma. Ez a 90 bájtnyi terület a VTOC tízes ($0A) bájtjából indul és a $63-ig tart. A bittérkép minden bitje egy szektort jelent. Az első bájt legbaloldalibb bitje a 0. szektorszámot, a következő bit az 1. szektort, az utolsó bájt legbaloldalibb bitje pedig a 719-es szektorszámot jelzi. Ha egy bit értéke 1, akkor a hozzá tartozó szektor az adott pillanatban nincs használatban, de rendel-

1. táblázat

	A-810	C-1541
Maximális fájl bejegyzés	64	144
Könyvtár (direktori) helye a lemezen	20. sáv	18. sáv
Egy könyvtári bejegyzés mérete	16 bájt	30 bájt
Könyvtári szektorok száma	8 (128 bájt/szektor)	(256 bájt/szektor)



2. ábra. Könyvtárszektorok

kezésre áll; ha értéke 0, akkor a szektor foglalt.

A *2. táblázatban* az ide vonatkozó információk alapján összehasonlítjuk az ATARI és a Commodore meghajtókat.

A C—1541 négybájtos bejegyzéséből az első bájt a sávban levő szabad szektorok számát adja meg. A három további bájton levő 24 bit az egyes szektorok foglaltságát jelzi. Ha az adott szektorra vonatkozó bit 0, akkor az foglalt, ha 1, akkor szabad. Mivel a sávon 24-nél kevesebb (változó számú) szektor van, ezért a nem létező szektoroknak 0 bit felel meg.

Adatszektorok

Az adatszektor a fájl adatbájtjait tartalmazza. 128 bájtból áll, amelyből 125 bájton adatok és három bájton kontroll információk foglalnak helyet *(4. ábra)*.

Az adatbájtok a 0. bájtnál kezdődnek és a 124. bájtig tartanak (beleértve a 124. bájtot is); kontroll információkat a 125., 126., 127. bájtok tartalmaznak.

A szektor bájtszámlálója a 125. bájton van. Ez az érték a felhasznált adatbájtok számát adja meg, 0-tól 125-ig terjedhet (0 = nincs adat). Ha nincs annyi érték, hogy a szektort teljes egészében kitöltse, akkor ez a szám kisebb, mint 125.

A 126. bájton belül balról az első 6 bit a fájl számát adja, értéke 0-tól 63-ig ($3F) terjedhet. A szám útmutatásul szolgál a fájlbejegyzés helyéhez a könyvtárba. A nullás fájlszám a $169. szektor nulladik fájlbejegyzése.

A 126-os bájton belül jobbról a két utolsó bit a 127. bájt nyolc bitjével együtt a fájl következő adatszektorára mutat. Ennek értéke nullától 719-ig terjedhet ($2CF). Ha ez az érték nulla, akkor nincs több szektor a fájl szektorláncában.

A fájl szektorláncának utolsó szektora a fájlvégszektor, az END OF FILE (EOF) szektor. Hogy az EOF szektor tartalmaz-e adatot, azt a szektor bájtszámláló értéke mutatja meg. Ha ez az érték nagyobb Ø-nál, akkor természetesen nem üres.

A C—1541-es meghajtónál a fájlszektorlánc-mutató a szektor első két bájtján van. A mutató első bájtja a sáv címét adja (értéke 1—35), a második bájt pedig a szektor címét (0-tól maximum 20-ig). Az utolsó adatszektor-mutató első bájtja 0, második bájtja a szektorban lévő adatbájtok számát adja meg.

A DOS működése

A gyártó cég a számítógéppel együtt felül nem írható rendszerlemezt is rendelkezésre bocsát. Ez tartalmazza a „lemezorientált operációs rendszert", a DOS-t. A C64 számítógépnél erre nincs szükség, mivel itt a DOS beépítve rendelkezésre áll.

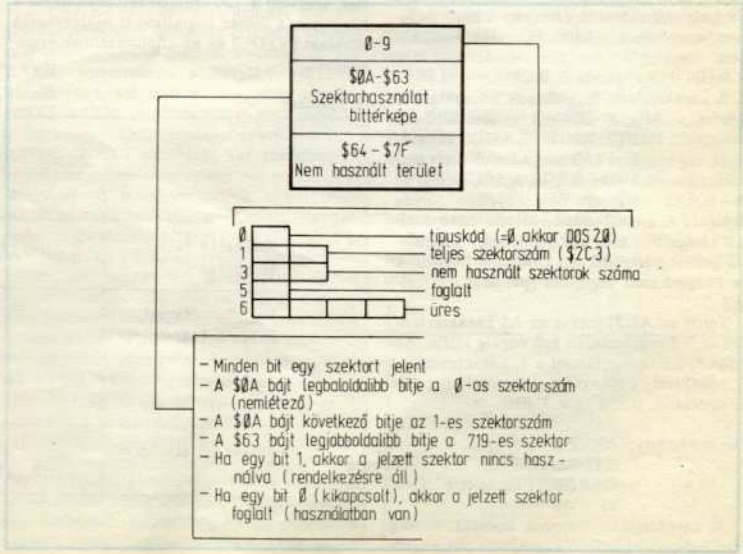
Célszerű a rendszerlemezről felhasználás előtt másolatot készíteni, és azt felülírás ellen levédeni, mivel így elkerülhetjük az eredeti lemez megrongálását. A számítógép bekapcsolása, a DOS kulcsszó bebillentyűzése, végül a RETURN lenyomása után a képernyőn megjelenik a DOS főmenü *(3. táblázat)*.

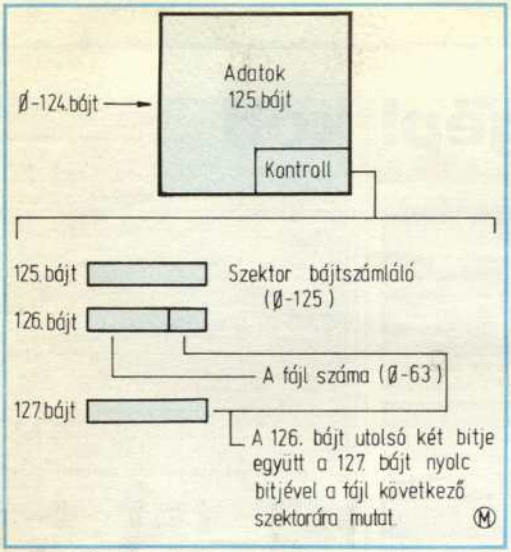
A menü a DOS 3 funkcióinak listája. A megfelelő első billentyük lenyomására működésbe lép a meghívott funkció. Minden funkció kérdés-felelet rendszerben dolgozik. A főmenü egészen addig visszahívható, míg a rendszerlemez a meghajtóban van, csupán az ESC vagy a RETURN billentyűt kell megnyomni.

A következőkben az egyes funkciók jelentését és használatát vizsgáljuk.

FILE INDEX. A lemez tartalomjegyzékét — azaz a lemezre vitt fájlok jegyzékét — hozza a képernyőre. A fájlok nevén kívül (max. 8 karakter) a fájlok által lefoglalt, valamint a szabad területet is tartalmazza.

3. ábra. VTOC szektor





4. ábra. Adatszektorok

	A-810 VTOC	C-1541 BAM
HELYE A LEMEZEN	$168 (354.) szektor	18. sáv, 0.szektor
HOSSZA	128 bájt (ebből a bit-térkép 90 bájt)	140 bájt
BEJEGYZÉS	90 bájt hosszon, folytonosan. Minden egyes bit egy szektort reprezentál.	4 bájt egy bejegyzés egy sávra vonatkozik: 4*35

2. táblázat

3. táblázat

FILE INDEX	LOAD	MEM SAVE
TO CARTRIDGE	SAVE	GO AT HEX ADDR
COPY/APPEND	ERASE	X-USER-DEFINED
DUPLICATE	RENAME	HELP
INIT DISC	PROTECT	
ACCESS DOS2	UNPROTECT	

TO CARTRIDGE. Meghivással kiléphetünk a DOS-ból és visszatérhetünk a szerkesztő módba.

COPY/APPEND. Fájlok másolását és összefüzését: egymás után másolását érhetjük el vele.

DUPLICATE. A DOS rendszerlemez másolása három részletben. Vigyázat, mert a lemezt, amelyre másolunk, újraformázza, tehát csak új lemezre készitsünk DUPLICATE másolatot, illetve olyanra, amely felülirható.

INIT DISC. Segitségével lemezt formázhatunk. Hasonlóan az előző funkciókhoz, a lemezen minden korábbi információ elvész.

ACCESS DOS2. Belépést jelent a DOS 2-be, tehát a DOS 2-ben irt lemezeket olvashatjuk e funkció meghivásával.

LOAD. A rendszerlemez könyvtárában levő programot (vagy a programok közül az általunk kiválasztottat) betölti a memóriába. Betöltés után — a felhasználó válaszától függően — a futást el is indithatja. Figyelem! A nem futtatható programokra kiadott futtatási parancsot csak a RESET-tel lehet hatástalanitani.

SAVE. Rendszerprogramok kimentését teszi lehetővé a rendszerlemezre. Minden esetben meg kell adni a fájl nevét, a kezdő- és végmemóriacimet, opcionálisan a kezdeti cimet, futtatási cimet, valamint az egységet.

ERASE. Segitségével programokat törölhetünk a lemezről.

RENAME. Meghivásával régi programoknak új nevet adhatunk.

PROTECT. Programok védelme törlés és átnevezés ellen. Az újraformázás és a duplikálás ellen azonban nem nyújt védelmet.

UNPROTECT. Az előző védelem feloldása.

MEM SAVE. A memória pillanatnyi állapotát menti ki lemezre.

GO AT HEX ADDR. Program indítása egy adott hexadecimális (tizenhatos számrendszerbeli) címről.

X-USER-DEFINED. A felhasználó által megadott — gépi kódban megirt — DOS szegmens definiálása.

HELP. Általános információt ad a menü egyes pontjainak használatáról.

A fájlvezérlési blokkok (FCB) szerepe

Az FMS fájlvezérlési blokkok a használatban levő fájlokra vonatkozó információ tárolására szolgálnak. Minden FMS által feldolgozás alatt álló fájl egy FCB-t követel meg. Az ATARI egyidejűleg 8 fájl kezelésére van felkészitve. Az FCB-k egy az egyben megfelelnek az IOCB-knek (Input-Output Control Block). Ha tehát egy fájl például 4 fájlszámmal kerül feldolgozásra, a fájlkezelő rendszer szintén a 4. FCB-t fogja hozzárendelni. Az FCB-k mérete azonos az IOCB-k méretével, azaz 16 bájt. Az FCB-k egy folytatólagos RAM területen helyezkednek el, akárcsak az IOCB-k. A fájlkezelő rendszer hivásakor az X regiszter tartalmazza az elmozdulást (az IOCB szám 16-szorosa) a felhivást megvalósító IOCB-hez képest. Az FMS arra használja ezt az elmozdulásértéket, hogy elérje mind az IOCB, mind az FCB információit. Az FCB a memória $1381. bájtján kezdődik. Az FCB nevezetes mutatói és értékeik az alábbiak.

FÁJLSZÁM. A feldolgozás alatt álló fájl száma. Ha egy fájl olvasásra lett megnyitva, feladata az adatszektorok egy fájlhoz tartozásának ellenőrzése. Íráskor ez az érték az adatszektorok fájlszámmezőjébe lesz bejegyezve.

MEGNYITÁSI TÍPUS KÓDJA. Mutató, amely a fájl megnyitási módjára utal:

INPUT $04
OUTPUT $08
UPDATE $0C
APPEND $01
DIRECTORY olvasás $02

DOS VERZIÓ MUTATÓ. A különböző DOS verziókban az adatszektorok hossza eltérő.

TERJESZKEDÉSI MUTATÓ. Ha értéke $80, a fájl alkalmas új adatszektorok elfoglalására. Ez az eset OUTPUT-ra vagy APPEND-re nyitott fájloknál. Ha az értéke $40, a kurrens szektor egy memóriapuffer, amely a módositás után visszairható a lemezre.

A MEGHAJTÓ TÍPUSAZONOSÍTÓJA. A különböző meghajtók különböző irássűrűséggel dolgoznak és az adatbájtok szektoronkénti száma eltérő.

ADATMUTATÓ. A soron következő feldolgozandó adatbájtra mutat. OUTPUT-ra vagy APPEND-re nyitott fájlnál a következő szabad bájtot az aktuális szektoron belül, UPDATE-nál a következő szektor elejét mutatja.

PUFFERINDEX. A szektorpuffer táblájában a puffercimek egyikére mutat. A feldolgozás alatt álló fájlban a szektorpuffer tartalmazza az adatszektorokat. Ez az index mutatja meg, hogy melyik puffer van az adott fájlhoz rendelve.

SZEKTORINDEX. Az aktuálisan a pufferben levő szektor száma.

A KÖVETKEZŐ SZEKTOR INDEXE. A feldolgozás alatt állót (a láncolás értelmében) követő szektor száma.

APPEND MUTATÓ. APPEND-re nyitott fájlnál az eredeti fájlhoz kapcsolandó szektorok kezdetére mutat. A kapcsolás az APPEND lezárásakor hajtódik végre. *(Folytatjuk)*

JÁNOSA—PAÁL—SÜTŐ

BASIC és gépi kód

Legutóbb az aritmetikai utasításokról volt szó; most egy korábbi feladat új megoldását ismerjük meg.

A programokról

Az 1986/5. számban jelentek meg azok a programok, amelyek a képernyőt csillagokkal írták tele, az időmérést illusztrálták, és bemutatták azt az esetet, amikor a gépi kód alkalmazásával a sebességnövekedés olyan minimális, hogy célszerű a BASIC-nél maradni. A gépi kódú változat működését az 1987/3. számban ismertettem.

Az egyes géptípusokon mért futási időket a *táblázatban* foglaltam össze. Az adatok hatvanad másodpercben értendők. Az 1. sor a BASIC változat futási idejét tartalmazza, a 2. és 3. sor a gépi kódú változatét. A 2. sor adatai magukban foglalják a rutinnak a DATA sorokból való betöltési idejét is; a 3. sorban csak a gépi kódú rész végrehajtási ideje szerepel. A VC20-ra vonatkozó időket a kisebb képernyőméretnek megfelelően módosított programokkal mértem.

sorsz.	VC20	C64	C16
1	62	142	205
2	22	36	60
3	9	22	45
4	185	441	513
5	35	42	45
6	1	2	2

A futási idők összehasonlító táblázata

A most közölt programok szintén csillagokkal írják tele a képernyőt — ezúttal feketékkel. Ennek semmi gyakorlati haszna nincs, de tanulságai miatt mégis érdemes a témával foglalkozni.

A megvalósítás módja közismert: a képernyőmátrixot feltöltjük a csillag karakter képernyőkódjával, 42-vel, a színmátrix megfelelő mezőibe pedig a fekete szín kódját töltjük.

Míg a korábbi programok (a VC20 eltérő képernyőméretétől eltekintve) mindhárom géptípuson egyformán futtathatók voltak, most a képernyő- és színmátrixok különböző elhelyezése miatt minden géphez más programra van szükség.

A BASIC program nem kíván magyarázatot. Három változata az *1/a, 1/b, 1/c listákon* látható. Lényeges eltérés csak a *d* változó értékében és a ciklushatárokban fedezhető fel. A *d* változóban a képernyőmátrix és a színmátrix egymásnak megfelelő mezőinek bájtban mért távolsága van.

```
100 rem vc20
110 tt=ti:d=30720:c=42:f=0
120 for i=7680 to 8185
130 poke i,c:poke i+d,f
140 next
150 tt=ti-tt
160 poke 198,0:wait 198,1:get a$
170 print chr$(147),tt
```

1/a lista

```
100 rem c64
110 tt=ti:d=54272:c=42:f=0
120 for i=1024 to 2023
130 poke i,c:poke i+d,f
140 next
150 tt=ti-tt
160 poke 198,0:wait 198,1:get a$
170 print chr$(147),tt
```

1/b lista

```
100 rem c16
110 tt=ti:d=1024:c=42:f=0
120 for i=2048 to 3047
130 poke i,f:poke i+d,c
140 next
150 tt=ti-tt
160 getkey a$
170 print chr$(147),tt
```

1/c lista

A ciklus lefutása után a program egy billentyű leütésére vár; a képernyő törlődik, majd megjelenik a végrehajtás ideje a belső óra időegységében, hatvanad másodpercben. Az így mért időket a táblázat 4. sorában láthatjuk.

A gépi kódú változat BASIC nyelvű betöltőprogramja a *2/a, 2/b, 2/c,* a disassembler kimenete a *3/a, 3/b, 3/c listákon* látható. A betöltőprogram elindítja az időmérést, majd a gépi kódú rutint. Az időmérés eredménye a táblázat 5. sorában van, és mint a programlistából kitűnik, tartalmazza a betöltés idejét is.

Nézzük meg a gépi kódú rész tiszta futási idejét. Ennek legegyszerűbb módja, ha beiktatjuk ezt a programsort: 205 TT=TI majd RUN 205 paranccsal indítjuk a programot. A képernyő egy pillanat alatt megtelik csillagokkal, és egy billentyű lenyomása után a letörölt képernyőn olvashatjuk a táblázat 6. sorában látható futási időt. Az eredmény — remélem — mindenkit meg-

2/a lista

```
100 rem vc20
110 tt=ti
120 for i=828 to 868 step 8
130 s=0
140 for j=0 to 7
150 read a : poke i+j,a : s=s+a
160 next
170 read a : if a<>s then 200
180 print "adathiba a" i "szamu sorban"
190 stop
200 next
210 sys 828 : tt=ti-tt
220 poke 198,0 : wait 198,1 :get a$
230 print chr$(147),tt
828 data 169,0,162,30,160,150,133,251,1055
836 data 134,252,133,253,132,254,162,23,1343
844 data 160,21,169,42,145,251,169,0,957
852 data 145,253,136,16,245,165,251,24,1235
860 data 105,22,133,251,133,253,144,4,1045
868 data 230,252,230,254,202,208,225,96,1697
```

```
100 rem c64
110 tt=ti
120 for i=828 to 868 step 8
130 s=0
140 for j=0 to 7
150 read a : poke i+j,a : s=s+a
160 next
170 read a : if a<>s then 200
180 print "adathiba a" i "szamu sorban"
190 stop
200 next
210 sys 828 : tt=ti-tt
220 poke 198,0 : wait 198,1 :get a$
230 print chr$(147),tt
828 data 169,0,162,4,160,216,133,251,1095
836 data 134,252,133,253,132,254,162,25,1345
844 data 160,39,169,42,145,251,169,0,975
852 data 145,253,136,16,245,165,251,24,1235
860 data 105,40,133,251,133,253,144,4,1063
868 data 230,252,230,254,202,208,225,96,1697
```

2/b lista

```
100 rem c16
110 tt=ti
120 for i=828 to 868 step 8
130 s=0
140 for j=0 to 7
150 read a : poke i+j,a : s=s+a
160 next
170 read a : if a<>s then 200
180 print "adathiba a" i "szamu sorban"
190 stop
200 next
210 sys 828 : tt=ti-tt
220 getkey a$
230 print chr$(147),tt
828 data 169,0,162,12,160,8,133,216,860
836 data 134,217,133,218,132,219,162,25,1240
844 data 160,39,169,42,145,216,169,0,940
852 data 145,218,136,16,245,165,216,24,1165
860 data 105,40,133,216,133,218,144,4,993
868 data 230,217,230,219,202,208,225,96,1627
```

2/c lista

3/a lista (VC20)

```
. 033c  a9 00     lda #$00
. 033e  a2 0c     ldx #$0c
. 0340  a0 08     ldy #$08
. 0342  85 d8     sta $d8
. 0344  86 d9     stx $d9
. 0346  85 da     sta $da
. 0348  84 db     sty $db
. 034a  a2 19     ldx #$19
. 034c  a0 27     ldy #$27
. 034e  a9 2a     lda #$2a
. 0350  91 d8     sta ($d8),y
. 0352  a9 00     lda #$00
. 0354  91 da     sta ($da),y
. 0356  88        dey
. 0357  10 f5     bpl $034e
. 0359  a5 d8     lda $d8
. 035b  18        clc
. 035c  69 28     adc #$28
. 035e  85 d8     sta $d8
. 0360  85 da     sta $da
. 0362  90 04     bcc $0368
. 0364  e6 d9     inc $d9
. 0366  e6 db     inc $db
. 0368  ca        dex
. 0369  d0 e1     bne $034c
. 036b  60        rts
```

```
033c  a900        lda #$00
033e  a204        ldx #$04
0340  a0d8        ldy #$d8
0342  85fb        sta $fb
0344  86fc        stx $fc
0346  85fd        sta $fd
0348  84fe        sty $fe
034a  a219        ldx #$19
034c  a027        ldy #$27
034e  a92a        lda #$2a
0350  91fb        sta ($fb),y
0352  a900        lda #$00
0354  91fd        sta ($fd),y
0356  88          dey
0357  10f5        bpl $034e
0359  a5fb        lda $fb
035b  18          clc
035c  6928        adc #$28
035e  85fb        sta $fb
0360  85fd        sta $fd
0362  9004        bcc $0368
0364  e6fc        inc $fc
0366  e6fe        inc $fe
0368  ca          dex
0369  d0e1        bne $034c
036b  60          rts
```

3/b lista (C64)

```
033c  a900        lda #$00
033e  a21e        ldx #$1e
0340  a096        ldy #$96
0342  85fb        sta $fb
0344  86fc        stx $fc
0346  85fd        sta $fd
0348  84fe        sty $fe
034a  a217        ldx #$17
034c  a015        ldy #$15
034e  a92a        lda #$2a
0350  91fb        sta ($fb),y
0352  a900        lda #$00
0354  91fd        sta ($fd),y
0356  88          dey
0357  10f5        bpl $034e
0359  a5fb        lda $fb
035b  18          clc
035c  6916        adc #$16
035e  85fb        sta $fb
0360  85fd        sta $fd
0362  9004        bcc $0368
0364  e6fc        inc $fc
0366  e6fe        inc $fe
0368  ca          dex
0369  d0e1        bne $034c
036b  60          rts
```

3/c lista (C16)

győz arról, hogy itt van értelme gépi kódban programozni.

A gépi kódú rutin működését a következő alkalommal elemezzük. Addig is érdemes a programlistákat tanulmányozni, esetleg a különböző változatokat összehasonlítani.

Néhány megjegyzés. Ha fekete helyett más színt választunk, a BASIC változat amúgy sem rövid végrehatási ideje kb. 7 százalékkal megnő. Érdekelheti az olvasókat, hogy miért. Gondolkozzanak el rajta, én mindenesetre magyarázatot a legközelebb adok.

Az itt közölt VC20-as programok 8 kbájtos vagy nagyobb tárbővítő használata esetén nem működnek. Az ezzel kapcsolatos problémák megoldásáról később lesz szó.

BARNA LÁSZLÓ

Z80 programok haladóknak Spectrumra és Primóra

E sorozat szándéka a segítségnyújtás azoknak, akik már megtanulták a Z80 programozását, és hosszabb program megírására vállalkoznak. Olyan programokat, rutinokat adok közre, amelyek jól szolgálnak nagyobb programokban (nagy tudású sprite-kezelő, sprite-definiáló, keskenykarakter-kivitel). A programlistákat igyekszem megjegyzésekkel és magyarázatokkal ellátni, de ezek helyenként csak utalások a működésre. Célom, hogy később olyan nagyobb programokat mutassak be, amelyek az előző rutinokat felhasználják.

A programok Spectrumra készültek, de sok közülük más gépeken is fut. A Primóra átíráshoz mindig megadom a szükséges változtatásokat.

Akinek bármilyen problémája, ötlete, megjegyzése van a programokkal kapcsolatban, levélben vagy személyesen megkereshet vele. A forrásszövegek a GENS3 assemblernek megfelelő formában vannak leírva. Azoknak a spectrumosoknak, akiknek nincs meg a GEN3, (korlátozott számban) kazettán elküldöm.

Gyors LDIR

A Z80 az LDIR és LDDR utasításaival jól támogatja a blokkmozgatást. E két utasítás végrehajtási ideje bájtonként 21 órajelciklus (a továbbiakban *T*). Ez általában elég gyors, de néha szükség van ennél gyorsabb blokkmásolásra — például olyan grafikai vagy játékprogramokban, ahol elkerülhetetlen a képernyő másolása.

Ez a program kívülről nézve ugyanazt csinálja, mint az LDIR, csak valamivel gyorsabban. Végrehajtási ideje kb. 17 *T* bájtonként, ha elég sok bájtot töltünk át. Tetszőleges programban minden LDIR kicserélhető CALL LDIRQ-ra, csak ne felejtsük el, hogy az akkumulátor tartalma megváltozik.

Az LDIRQ programmal a Spectrumon egy 6 kbájtos blokkot (például a képernyőt) 30 ms alatt másolhatunk, szemben az eredeti 32 ms-mal. Primón a két érték 42 és 52 ms, a kisebb órajelfrekvencia miatt. Az időnyereség kb. 20 százalék.

A program működésének lényege a következő. Az LDI utasítás egy bájtot mozgat, és ez neki 16 *T* ideig tart. Ha elég sokat írunk le belőle egymás után, és ciklusba szervezzük őket, a sebesség a 16 *T*/bájthoz fog közelíteni (32 darab elég). A ciklus 32 bájtot 550 *T* ideig mozgat, ami 17,2 *T*/bájt. A ciklus lefutása után visszamaradó, 32-nél kevesebb bájtot a „közönséges" LDIR-rel mozgatjuk.

Senki sem szeret 32 egyforma utasítást egymás után bepötyögtetni. Ezért a program első lefutásakor elhelyezi önmagában a 31 hiányzó LDI-t, és ezzel ezt az öngenerálót is javarészt felülírja. Így nem sokkal hosszabb a program, a forrásszöveg viszont 19 sorral rövidebb. Az első lefutás alig lassúbb a többinél (1415 *T*), hiszen csak 62 bájtot mozgat a megadotton felül.

A programból a következőképpen csinálhatunk gyors LDDR-t:

— a 7. sorban LDI helyett LDD kell,

— a 25. sorban LDIR helyett LDDR-t írunk,

— a címkékben az I betűket a rend kedvéért cseréljük ki D-re.

Ha csak 32-vel osztható hosszúságú blokkokat akarunk mozgatni, a *-gal jelzett sorok elhagyhatók.

UHERKOVICH PÉTER
Bp., Irinyi út 42. 1117

```
 1     ;-----------------------------------;
 2     ;                                   ;
 3     ;     GYORS LDIR  UHI 1986          ;
 4     ;                                   ;
 5     ;-----------------------------------;
 6 LDIRQ   JP      LDIRQ3
 7 LDIRQ1  LDI                 ;EGY A 32-BOL
 8 LDIRQ2  PUSH    BC          ;ITT KEZDODIK
 9         PUSH    DE          ;ONMAGAT FELUL
10         PUSH    HL          ;IRO RESZ
11         LD      HL,LDIRQ1   ;ELSO LDI CIME
12         LD      DE,LDIRQ2   ;KOVETKEZO CIML
13         LD      BC,62       ;31 LDIR HOSSZA
14         JP      CREATE      ;62 BAJT
15         DEFS    47          ;IDE IS LDI JON
16 LDIRQ3  LD      A,B         ;ELLENORZI,HOGY
17         OR      A           ;VAN-E MEG LEG-
18         JR      NZ,LDIRQ1   ;ALABB 32 BAJT.
19         LD      A,C
20         CP      32
21         JR      NC,LDIRQ1   ;VAN MEG...
22         OR      A           ;(*)
23         RET     Z           ;(*)
24         LDIR                ;(*)  MARADEKOT
25         RET                 ;     TOLTI AT
26 CREATE  LDIR
27         POP     HL          ;A 31 HIANYZO
28         POP     DE          ;LDI BETOLTESE
29         POP     BC          ;MAJD AZ LDIRQ
30         JP      LDIRQ       ;FOLYTATASA
```

Teknősbéka-grafika III.

Eddigi parancskészletünket ki kell egészítenünk néhány további alapparanccsal. Első rajzunk elkészítése után, amint újat akarunk kezdeni, rájövünk, hogy nem tudjuk a régit letörölni.

Képernyőtörlés, középre állítás és ezek kombinációjára szolgáló parancsokat (CLEAN, HOME, CLEARSCREEN) kell tehát beiktatnunk, ami a parancskészlet-kiíratható rész (20-as, 30-as sor) kiegészítését és egy másik feldolgozó rész (170—190-es sorok) hozzáírását jelenti. Mindkettő igen egyszerű, nem tartalmaz semmilyen új, ismeretlen utasítást.

Eddig általában egy-egy sor betoldásával sikerült parancskészletünket kiegészítenünk; az egyetlen kivétel a LOGO nyelvben nem használatos képernyőnyomtató parancs volt. A most következő REPEAT parancs beiktatása azonban több munkát igényel.

Ennél a parancsnál meg kell adnunk az ismétlések számát, az ismétlendő részt, majd ezt végre kell hajtanunk. A 20-as sor kiegészítésén kívül (melyben a [helyett Ä,] helyett Å karaktereket írt a nyomtató a *listában* és az *1. ábrán*), először a 80-as, 90-es sorokban meg kell vizsgálnunk, volt-e REPEAT parancs. Ha igen, akkor a feldolgozandó parancssorozatból el kell különítenünk a még hátralévő részt (A$), és át kell ugranunk a feldolgozó szubrutinra. A 350-es sorban adjuk meg az ismétlések számát (CC), és elkezdjük az ismétlendő rész meghatározását a II ciklussal. A 360-as és 370-es sorban a 100-120-as sorokéhoz hasonló módszerrel jelöljük ki az ismétlendő rész végét (az elejét a 350-es sorban I+1-re állítottuk).

A 380-as sorban megkezdjük az ismétlések (II) és az ismétlendő rész (I1) feldolgozását.

Maga az ismétlendő rész feldolgozása (390—620-as sorok) megegyezik a parancssorozatéval (100—330-as sorok). A befejezés (630-as sor) itt csak azért más, mert ismételni is kell. A *2. ábrán* látható képet az alábbi parancssorozattal állítottuk elő:

REPEAT1Ø[FD5Ø;RT9Ø;FD5Ø;RT9Ø;
FD5Ø;RT9Ø;FD5Ø;RT9Ø;RT36

Láthatjuk, hogy az ismétlendő részen belül nincs a nyomtatásra vagy a REPEAT parancsra vonatkozó vizsgálat. Ilyen itt nem is lehet. A REPEAT parancs után újabb REPEAT vagy nyomtatóparancs azonban már következhet.

Mit eredményez ez? A 2. ábrát előállító parancssorozatot a LOGO nyelvben a következő, rövidebb sorozattal helyettesíthetjük:

REPEAT1Ø[REPEAT4[FD5Ø;RT9Ø]RT36]

Itt ez azonban hibás eredményt ad. Nézzük meg, hogy miért. A program megtalálva az első REPEAT parancsot, elhagyja az eddigi részt, tehát A$ = A1$, mivel I = 1. Meghatározza az ismétlések számát: CC = 1Ø, mivel LEN(A$) = 32, és az első két vizsgált karaktert nem szám követi, ezért a VAL függvény csak ezt a két karaktert alakítja át számmá. Eddig minden rendben van. Az ismétlendő rész végének meghatározása azonban már hibás eredményre vezet, hiszen az első megtalált] jel a második REPEAT parancshoz tartozik. Az ismétlésből

```
PARANCSOK:
FD-ELOERE
BK-HAATRA
LT-BALRA
RT-JOBBRA
MINDEGYIK UTAAN EGY SZAAM AALL.
PU-TOLL FEL
REPEATA...A-ISMEETLEES
Ü-NYOMTATAAS
CS-KEEPTOERLEES, KOEZEEPRE
HOME-KOEZEEPRE
CLEAN-KEEPTOERLEES
A PARANCSOK SORDZATBA KAPCSOLHATOOK
AZ ELVAALASZTOJEL A PONTOSVESSZOE.
```

1. ábra

```
10 REM RAJZOLO III
20 PRINT"PARANCSOK:":PRINT"FD-ELOERE":PR
INT"BK-HAATRA":PRINT"LT-BALRA":PRINT"RT-
JOBBRA":PRINT"MINDEGYIK UTAAN EGY SZAAM
AALL.":PRINT"PU-TOLL FEL":PRINT"PD-TOLL
LE":PRINT"REPEATA...A-ISMEETLEES":PRINT"
Ü-NYOMTATAAS"
30 PRINT"CS-KEEPTOERLEES, KOEZEEPRE":PRI
NT"HO-KOEZEEPRE":PRINT"CL-KEEPTOERLEES":
PRINT"A PARANCSOK SOROZATBA KAPCSOLHATOO
K":PRINT"AZ ELVAALASZTOJEL A PONTOSVESSZ
OE."
40 CLEAR2000:X0=128:Y0=96:PI=355/113:ALF
A=PI/2:FI=ALFA:X=X0:Y=Y0:PMODE4,1:PCLS:R
AJZ$="M =X;, =Y;"
50 B$=INKEY$:IFB$="" THEN50 ELSE INPUTA1
$:A=LEN(A1$)
60 IFA1$="Ü" THEN GOSUB640:STOP
70 FOR I=1TO A:J=0
80 IF MID$(A1$,I,6)<>"REPEAT" THEN100
90 A$=RIGHT$(A1$,LEN(A1$)-I+1):GOSUB350:
GOTO340
100 J=J+1:IF J=A-I+1 THEN120
110 J$=MID$(A1$,I-1+J,1):IF J$<>"]" THEN
100
120 A$=MID$(A1$,I,J):I=I+J-1
130 IF J$="]" THEN A$=LEFT$(A$,LEN(A$)-1
)
140 IF FI>2*PI THEN FI=FI-(2*PI)
150 C$=LEFT$(A$,2):IF (A$="PU")AND(LEN(R
AJZ$)=10) THEN RAJZ$="B"+RAJZ$:GOTO310
160 IF (A$="PD")AND(LEN(RAJZ$)=11) THEN
RAJZ$=RIGHT$(RAJZ$,LEN(RAJZ$)-1):GOTO310
170 IF C$="CS" THEN PCLS:X=128:Y=96:GOTO
340
180 IF C$="HO" THEN X=128:Y=96:GOTO340
190 IF C$="CL" THEN PCLS:GOTO340
200 D$=MID$(A$,3,LEN(A$)-2):D=VAL(D$):SC
REEN1,1:IF (C$="FD")OR(C$="BK") THEN270
210 IF C$="LT" THEN250
220 IF C$="RT" THEN240
230 GOTO310
240 D=360-D
250 FI=FI+(D*PI/180)
260 GOTO310
270 IF C$="BK" THEN D=-D
280 X=X+(D*COS(FI)):Y=Y-(D*SIN(FI))
290 IF(X>-1)AND(X<256)AND(Y>-1)AND(Y<192
) THEN DRAW RAJZ$:GOTO340
300 IFX<0THEN X=0
310 IFX>255THEN X=255
320 IFY<0THEN Y=0
330 IFY>191THEN Y=191
340 NEXTI:GOTO50
350 CC=VAL(MID$(A$,7,LEN(A$)-6)):I=I+1:J
=0:FOR II=I TO A
360 J=J+1:J$=MID$(A1$,II+J,1):IFJ$<>"A"
THEN360
370 A2$=MID$(A1$,I+5+LEN(STR$(CC)),J-5-L
EN(STR$(CC))):I=I+J:A2=LEN(A2$)
380 FORII=1 TO CC:FOR I1=1 TO A2:J=0
390 J=J+1:IF J=A2-I1+1 THEN410
400 J$=MID$(A2$,I1-1+J,1):IF J$<>"]" THE
N390
410 A$=MID$(A2$,I1,J):I1=I1+J-1
420 IF J$="]" THEN A$=LEFT$(A$,LEN(A$)-1
)
430 IF FI>2*PI THEN FI=FI-(2*PI)
440 C$=LEFT$(A$,2):IF (A$="PU")AND(LEN(R
AJZ$)=10) THEN RAJZ$="B"+RAJZ$:GOTO630
450 IF (A$="PD")AND(LEN(RAJZ$)=11) THEN
RAJZ$=RIGHT$(RAJZ$,LEN(RAJZ$)-1):GOTO630
460 IF C$="CS" THEN PCLS:X=128:Y=96:GOTO
630
470 IF C$="HO" THEN X=128:Y=96:GOTO630
480 IF C$="CL" THEN PCLS:GOTO630
490 D$=MID$(A$,3,LEN(A$)-2):D=VAL(D$):SC
REEN1,1:IF (C$="FD")OR(C$="BK") THEN560
500 IF C$="LT" THEN540
510 IF C$="RT" THEN530
520 GOTO630
530 D=360-D
540 FI=FI+(D*PI/180)
550 GOTO630
560 IF C$="BK" THEN FI=FI+PI
570 X=X+(D*COS(FI)):Y=Y-(D*SIN(FI))
580 IF (X>-1)AND(X<256)AND(Y>-1)AND(Y<19
2) THEN DRAW RAJZ$:GOTO630
590 IF X<0THEN X=0
600 IF X>255THEN X=255
610 IF Y<0THEN Y=0
620 IF Y>191THEN Y=191
630 NEXT I1:NEXT II:RETURN
640 PRINT#-2,CHR$(13);CHR$(24);CHR$(27);
"Q";CHR$(27);"T08";:AA=1536:FORII=0TO191
STEP4:DD=32*II+AA:FORJJ=0TO31:Y1(0)=PEEK
(JJ+DD):Y2(0)=PEEK(JJ+32+DD):Y3(0)=PEEK(
JJ+64+DD):Y4(0)=PEEK(JJ+96+DD):C1=256:FO
RKK=1TO8:C1=C1/2:B1=C1*INT(Y1(KK-1)/C1)
650 Y1(KK)=Y1(KK-1)-B1:B2=C1*INT(Y2(KK-1
)/C1):Y2(KK)=Y2(KK-1)-B2:B3=C1*INT(Y3(KK
-1)/C1):Y3(KK)=Y3(KK-1)-B3:B4=C1*INT(Y4(
KK-1)/C1):Y4(KK)=Y4(KK-1)-B4:PRINT#-2,CH
R$(27);"W0008";:IFB1<>0THEN EE=15:GOTO66
0 ELSE EE=0:GOTO660
660 IFB2<>0THEN PRINT#-2,CHR$(EE+240);:G
OTO670 ELSE PRINT#-2,CHR$(EE);:GOTO670
670 IFB3<>0THEN EE=15:GOTO680 ELSE EE=0:
GOTO680
680 IFB4<>0THEN PRINT#-2,CHR$(EE+240);:G
OTO690 ELSE PRINT#-2,CHR$(EE);:GOTO690
690 NEXT KK:NEXT JJ:PRINT#-2,CHR$(13);:N
EXT II:RETURN
```

Adatbeolvasás Pascalban

ZX–SPECTRUM

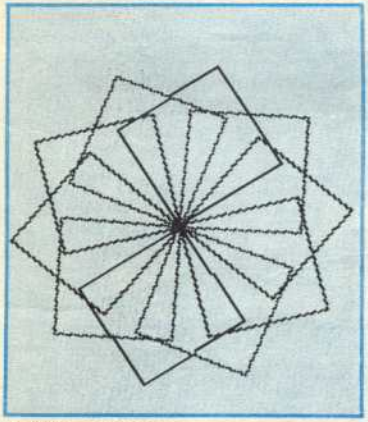
tehát el fog maradni az utolsó parancs. Mivel így

A2$="REPEAT4[FD5Ø;RTØ"

lesz, ezért további hibákkal is számolnunk kell.

A feldolgozásnál az első ; jelig terjedő részt egyetlen parancsnak veszi. Így a 420-as sorban

A$="REPEAT4[FD5Ø"



2. ábra

lesz a feldolgozandó parancs. Ebből leválasztva az első két karaktert, C$="RE" adódik, ami nem azonos egyik értelmes paranccsal sem. A 490-es sorban, mivel a harmadik karakter nem szám, D=Ø lesz, és az 550-es sorban továbbugrik anélkül, hogy a parancs hatására valami is történne. Ezután rátér a következő parancs vizsgálatára, amit jól végre is hajt. Így az első ciklusból tízszeri 90°-os jobbra fordulás lesz, vagyis teljesen más, mint amit akartunk.

Programunkkal tehát feldolgozhatunk ún. egyszintű REPEAT parancsokat tartalmazó parancssorozatot, de ún. egymásba ágyazottakat nem. Eddigi módszerünk, hogy újabb parancsokat sorok, sorrészek hozzáadásával építhetünk be programunkba, itt már nem ad helyes eredményt. Eljutottunk oda, ahol a legtöbb esetben a fejlesztők is megakadnak: foldozgatással már nem boldogulunk. Teljes szerkezeti átalakításra van szükség.

DR. SIMONYI ENDRE

Akik a programozást BASIC nyelven tanulták, hozzászoktak ahhoz, hogy ez a nyelv messzemenően támogatja az interaktív programok készítését. Ha ezután — a BASIC korlátait látva — áttérnek a Pascalra, keserű csalódások forrása lehet az, hogy a nyelv eredeti eszközkészlete ugyan lehetővé tesz nagyon sok mindent, de legtöbbször azon az áron, hogy nekünk magunknak kell a korábban már megszokott szolgáltatásokat megvalósító programrészeket is megírni.

Ez a helyzet például a hatványozással, amely művelet az elemi aritmetikai utasításokkal megfogalmazható, ezért kihagyták a Pascal eszköztárából; de ilyen a beolvasási műveletek BASIC-ben megszokott „bolondállósága", vagyis az a jóindulatú sajátsága, hogy az adatok beolvasásánál elkövetett szintaktikai hibák javíthatók, nem okozzák a program visszavonhatatlan megszakítását. Ez a javíthatóság az interaktív programok nagyon is lényeges követelménye, hiszen ellenkező esetben gyakorlatilag nem készíthető olyan program, amely sok beolvasott adattal dolgozik, mert mindig lehet számítani a felhasználó tévesztésére.

A Pascal standard adatbeolvasó eszköze — a READ és a READLN függvény — sajnos olyan, hogy ha nem a típusnak megfelelő adatot olvasunk be, akkor hibaüzenettel a program futása véglegesen megszakad. Ilyen hiba lehet például egy valós szám beolvasásakor, ha véletlenül olyan billentyűt nyomunk meg, amelyhez tartozó karakter nem fordul elő a számformátumban: például betű az „E" kivételével vagy vesszőt a tizedespont helyett, továbbá ha megfeledkezünk arról, hogy a Pascalban a szám mindig számjeggyel vagy előjellel kezdődik, és a BASIC-ben megszokott lehetőséggel élve tizedesponttal akarjuk kezdeni a számot. Ugyancsak visszavonhatatlan hibát okoz a legnagyobb ábrázolható számnál nagyobb szám beolvasásának kísérlete is.

A következőkben megmutatjuk, hogy a ZX—Sprectrumra készített HISOFT Pascal HP4T fordítóprogramra hogyan lehet olyan eljárásokat írni, amelyek a fenti hibákat nemcsak kiküszöbölik, hanem a BASIC-ben megszokott szolgáltatásokat meg is haladják. Itt arra gondolunk, hogy elegánsabb programok készíthetők, ha a beolvasott szöveg nem a BASIC módszere szerint a képernyő alsó részének bemeneti „ablakán" át érkezik, sem pedig az eredeti Pascal READ függvény szerint ott, ahol a kurzor éppen áll, hanem a képernyőnek a programozó által megadott tetszőleges helyén.

A példaprogram a ReadReal és a ReadInteger függvényekkel a képernyőre rendezett formában olvas be soronként egy egész típusú és két valós típusú számot. A program sok hasznosat nem csinál, de arra lehetőséget ad, hogy megfigyeljük, hogyan lehet saját programjainkban alkalmazni az új bevitel lehetőségeit. A két függvény nem enged szintaktikailag hibás számot beolvasni, a beolvasást addig ismétli, amíg hibátlan bemenő adatot nem kap.

A függvényeket a Spectrum-felhasználók minden változtatás nélkül beépíthetik saját programjaikba, de más gépen dolgozók számára is hasznosak, mivel az alapgondolat minden Pascal implementációban megvalósítható. A ReadReal függvény ötletét maga Jensen és Wirth adta „A Pascal programozási nyelv. Felhasználói kézikönyv és a nyelv formális leírása" c. könyvének F. függelékében. Itt ugyanis közlik rdr eljárásnév alatt a Standard Pascal read eljárásának forrásnyelvű listáját. Nos, mi ezt az eljárást dolgoztuk fel, természetesen a Spectrum és a HP4T sajátosságaihoz igazítva, feladatunk megoldásához.

A függvény működésének lényege, hogy egy 20 karakterből álló ST nevű tömbbe beolvassuk a kívánt bemenő adatot: valós számot (de sztringként!), majd a program elemezni kezdi a sztringet, hogy a valós szám szintaktikájának megfelel-e. Ha igen, az elemzéssel párhuzamosan átírja valós számmá, valahogy úgy, ahogy a BASIC VAL függvénye tenné. Ha valahol szintaktikai hibára bukkan a program, akkor abbahagyja a további vizsgálódást, törli a képernyőre kiírt sztringet (hibás számot), és újabb beolvasásra vár.

```
B7B6   20 PROGRAM InputDemo;
B7B6   30 (1986. szept. 27. Kaboláy)
B7B6   40 VAR A:REAL;
B7BF   50     I,J,B:INTEGER;
B7BF   60
B7BF   70 PROCEDURE SPOUT(C:CHAR);
B7C2   80 BEGIN
B7DA   90   INLINE($FD,$21,$3A,$5C,
B7DE  100            $DD,$7E,2,$D7)
B7E2  110 END;{SPOUT}
B7E9  120
B7E9  130 PROCEDURE AT(x,y:INTEGER);
B7EC  140 BEGIN
B804  150   SPOUT(CHR(22));
B813  160   SPOUT(CHR(x));
B825  170   SPOUT(CHR(y))
B82E  180 END;{AT}
B841  190
B841  200 FUNCTION ReadReal(Sor,Oszlop:INTEGER):REAL;
B844  210 {REAL szamot olvas be a Sor, Oszlop karakter pozicioba}
B844  220 LABEL 13;
B84A  230 CONST t23=8388608;  {2^23}
B84A  240       limit=1677721;{t DIV 5}
B84A  250       z=48;         {ORD('0')}
B84A  260       limi=32;      {max. kitevo}
B84A  270       limi2=-32;    {min. kitevo}
B84A  280       hossz=8;
B84A  290 TYPE  posint=0..32;
B84A  300 VAR ch,NULL:CHAR;
B84A  310     a,y:REAL;
B84A  320     i,j,e:INTEGER;
B84A  330     q,s,ss:BOOLEAN;
B84A  340     ST:ARRAY[1..20]OF CHAR;
B84A  350
B84A  360 PROCEDURE GET(VAR C:CHAR);
B84D  370 BEGIN
B865  380   j:=j+1;
B882  390   IF j<=20 THEN C:=ST[j];
B8D3  400   IF (ORD(C)=0)AND(j>0)THEN j:=j-1;
B925  410   IF j=0 THEN C:='A'
B94B  420 END;{GET}
B953  430
B953  440 FUNCTION ten(e:posint):REAL;{=10^e, 0<e<32}
B956  450 VAR i:INTEGER;
B956  460     t:REAL;
B956  470 BEGIN
B96E  480   i:=0;
B977  490   t:=1;
B989  500   REPEAT
B989  510     IF ODD(e) THEN
B999  520       CASE i OF
B99F  530         0:t:=t*1E1;
B9CF  540         1:t:=t*1E2;
B9FF  550         2:t:=t*1E4;
BA2F  560         3:t:=t*1E8;
BA5F  570         4:t:=t*1E16;
BA8F  580         5:t:=t*1E32
BAAF  590       END;
BABC  600     e:=e DIV 2;
BAD0  610     i:=i+1
BADB  620   UNTIL e=0;
BAF2  630   ten:=t
BAF2  640 END;{ten}
BB16  650
BB16  660 PROCEDURE SIGN(VAR a:BOOLEAN);
BB19  670 BEGIN
BB31  680   GET(ch);
BB44  690   IF ch='-'THEN
BB5C  700   BEGIN
BB5C  710     a:=TRUE;
BB67  720     GET(ch)
BB71  730   END ELSE
BB7D  740   BEGIN
BB7D  750     a:=FALSE;
BB87  760     IF ch='+' THEN GET(ch)
BBA9  770   END
BBB2  780 END;{SIGN}
BBB9  790
BBB9  800 BEGIN
BBD1  810  ch:=' ';
BBD6  820   REPEAT
BBD6  830     j:=0;
BBE2  840     AT(Sor,Oszlop+hossz);
BC02  850     FOR i:=1 TO hossz DO
BC25  860     WRITE(CHR(8));
BC32  870     a:=TRUE;
BC37  880     READLN;
BC3A  890     READ(ST);
BC46  900     SIGN(s);
BC53  910     IF NOT (ch IN['0'..'9'])THEN
BC88  920     BEGIN
BC88  930       q:=FALSE;
```

```
BC8C  940       GOTO 13
BC8F  950     END;
BC8F  960     a:=0;
BCA1  970     e:=0;
BCAA  980     REPEAT{Egeszresz beolvasasa}
BCAA  990       IF a<limit THEN a:=10*a+ORD(ch)-z
BCFF 1000       ELSE e:=e+1;
BD20 1010       GET(ch)
BD28 1020     UNTIL NOT(ch IN['0'..'9']);
BD62 1030     IF ch='.'THEN
BD73 1040     {Tortresz beolvasasa}
BD73 1050     BEGIN
BD73 1060       GET(ch);
BD80 1070       WHILE ch IN['0'..'9']DO
BDB6 1080       BEGIN
BDB6 1090         IF a<limit THEN
BDD9 1100         BEGIN
BDD9 1110           a:=10*a+ORD(ch)-z;
BE19 1120           e:=e-1
BE24 1130         END;
BE26 1140         GET(ch)
BE2E 1150       END
BE33 1160     END;
BE34 1170     IF (ch='e')OR(ch='E')THEN
BE57 1180     {Kitevo olvasas}
BE57 1190     BEGIN
BE57 1200       i:=0;
BE60 1210       SIGN(ss);
BE6D 1220       IF ch IN['0'..'9']THEN
BEA0 1230       BEGIN
BEA0 1240         i:=ORD(ch)-z;
BEB5 1250         GET(ch);
BEC2 1260         WHILE ch IN['0'..'9']DO
BEF8 1270         BEGIN
BEF8 1280           IF i<limit THEN i:=10*i+ORD(ch)-z;
BF4C 1290           GET(ch)
BF54 1300         END
BF59 1310       END ELSE
BF5F 1320       BEGIN
BF5F 1330         q:=FALSE;
BF63 1340         GOTO 13
BF66 1350       END;
BF66 1360       IF ss THEN e:=e-i
BF74 1370       ELSE e:=e+i
BF92 1380     END;
BFA5 1390     IF e<limi THEN
BFBC 1400     BEGIN
BFBC 1410       a:=0;
BFCE 1420       e:=0
BFD2 1430     END ELSE
BFDA 1440     IF e>limi THEN
BFF0 1450     BEGIN
BFF0 1460       q:=FALSE;
BFF4 1470       GOTO 13
BFF7 1480     END;
BFF7 1490     {0<a<2^23}
BFF7 1500     y:=a;
C00F 1510     IF s THEN y:=-y;
C036 1520     IF e<0 THEN y:=y/ten(-e)
C067 1530     ELSE IF e<>0 THEN y:=y*ten(e);
C0C2 1540 13:UNTIL q=TRUE;
C0D3 1550     ReadReal:=y;
C0EB 1560     AT(Sor,Oszlop+hossz);
C108 1570     FOR i:=0 TO hossz-j DO
C141 1580     WRITE(CHR(8))
C14B 1590 END;{ReadReal}
C15E 1600
C15E 1610 FUNCTION ReadInteger(Sor,Oszlop:INTEGER):INTEGER;
C161 1620 {INTEGER szam beolvasasa Sor, Oszlop karakter pozicioba}
C161 1630 VAR a:REAL;
C161 1640 BEGIN
C179 1650   WHILE (ABS(a))>MAXINT) DO
C1A9 1660     a:=ReadReal(Sor,Oszlop);
C1D3 1670   WHILE NOT (a=TRUNC(a)) DO
C20A 1680     a:=ReadReal(Sor,Oszlop);
C234 1690   ReadInteger:=TRUNC(a)
C242 1700 END;{ReadInteger}
C255 1710
C255 1720 BEGIN
C25E 1730   PAGE;
C263 1740   AT(0,0);
C270 1750   WRITELN('Integer    Real       Real');
C296 1760   WRITELN;
C299 1770   FOR I:=0 TO 16 DO
C2B3 1780   BEGIN
C2B6 1790     B:=ReadInteger(I*2,0);
C2CA 1800     FOR J:=0 TO 1 DO
C2E4 1810       A:=ReadReal(I*2,J*10+10)
C302 1820   END
C313 1830 END.
End Address: C316
Run?C
```

Ismerkedjünk meg a vázolt algoritmus részleteivel is! Az ST sztring egy-egy karakterének vizsgálatát a ReadReal függvényben definiált GET eljárás kezdi. Ez az eljárás nem tesz mást, csak számolja a megvizsgált karaktereket, nehogy a sztringből kifusson (j-nek 0-nál nagyobbnak és 20-nál kisebbnek kell lennie!). Ha beolvasás közben töröltünk (CHR 8), akkor ezt is számontartja.

Az előjelkarakter vizsgálatára külön eljárást deklaráltunk SIGN néven, melyet a program kétszer hív: először a mantissza, majd a karakterisztika előjelének vizsgálatakor. A SIGN eljárásból visszatérve az s logikai változó az előjeleknek megfelelő értéket kap, az aktuális karakterváltozónak pedig számnak kell lennie. Ha mégsem az, a vizsgálat leáll: a program a 13. címkére ugrik q=FALSE értékkel, ami a hibás sztringet törli és az eljárást megismétli.

```
      B
  10 INPUT "n=";n
  20 DIM a(n,n+1)
  30 PRINT "Egyutthatok:"
  40 FOR i=1 TO n
  50 FOR j=1 TO n
  60 PRINT i;". sor, ";j;". elem: ";
  70 INPUT a(i,j)
  80 PRINT a(i,j)
  90 NEXT j
 100 PRINT
 110 PRINT "i";".sor jobboldal: ";
 120 INPUT a(i,n+1)
 130 PRINT a(i,n+1)
 140 PRINT
 150 NEXT i
1000 FOR i=1  TO n
1010 LET m=1/a(i,i)
1020 FOR k=i+1 TO n+1
1030 LET a(i,k)=m*a(i,k)
1040 NEXT k
1050 FOR j=1 TO n
1060 IF j=i THEN  GO TO 1100
1070 LET T=a(j,i)
1080 FOR k=i+1 TO n+1
1090 LET a(j,k)=a(j,k)-T*a(i,k)
1100 NEXT k
1110 NEXT j
1120 NEXT i
2000 CLS : PRINT "EREDMENYEK"
2010 FOR i=1 TO n
2020 PRINT i;TAB 4;a(i,n+1)
2030 NEXT i
```

Ezután a valós szám egész részének beolvasása következik, és egyidejűleg az *a* REAL típusú segédváltozóban képezzük a már beolvasott számjegyekből a mantissza értékét. (Minden alkalommal *a* értékét 10-zel szorozzuk és hozzáadjuk az újonnan kiértékelt számjegyet.) Ha a tizedespontot is beolvasta a program, a fenti algoritmus folytatódik: *a*-ban most már az egész mantissza tárolódik egész szám formájában. A decimális törtszámjegyek számát a program az *e* INTEGER változóban jegyzi a művelet alatt.

Ha a karakterek beolvasása során a program „e"-re vagy „E"-re akad, felismeri, hogy a mantissza beolvasása befejeződött, és a karakterisztika következik. Ismét megvizsgálja az előjelet a SIGN függvénnyel, majd a karakterisztika számjegyeinek kiértékelése kezdődik el a fent leírtakhoz hasonlóan. Az új karakterek olvasása addig tart, amíg nem számjegy karakterre bukkan a program. Ekkor megvizsgálja, hogy a karakterisztika az ábrázolható legnagyobb értéket nem lépi-e túl. Ha nem, akkor a program a ten eljárás segítségével kiértékeli a szám nagyságrendjét és megszorozza a mantisszával, vagyis kiszámítja a beolvasott számot.

Ha az eljárás során a program bárhol hibás karaktert talál, ugrás következik be a 13. címkére a *q* logikai változó egyidejű FALSE értéke mellett. Ez az eljárás megismétlését váltja ki. Ha viszont a beolvasás hibátlan, a ReadReal függvény értéke felveszi az *y* munkaváltozó értékét, és az algoritmus a főprogramban folytatódik.

A programban deklaráltuk a SPOUT gépi kódú eljárást abból a célból, hogy a Spectrum eredeti BASIC kiíró rutinját

```
B7C4   20 PROGRAM InputDemo;
B7C4   30 (1986. szept. 27. Kaboldy)
B7C4   40 VAR A:REAL;
B7CD   50     I,J,B:INTEGER;
B7CD   60
B7CD   70 PROCEDURE SPOUT(C:CHAR);
B7D0   80 BEGIN
B7EB   90   INLINE(/FD,/21,/3A,/5C,
B7EC  100         /DD,/7E,2,/D7)
B7F0  110 END;(SPOUT)
B7F7  120
B7F7  130 PROCEDURE AT(x,y:INTEGER);
B7FA  140 BEGIN
B812  150   SPOUT(CHR(22));
B821  160   SPOUT(CHR(x));
B833  170   SPOUT(CHR(y))
B83C  180 END;(AT)
B84F  190
B84F  200 FUNCTION ReadReal(Sor,Oszlop:INTEGER):REAL;
B852  210 (REAL szamot olvas be a Sor, Oszlop karakter poziciora)
B852  220 LABEL 13;
B858  230 CONST t23=8388608;  (2^23)
B858  240       limit=1677721;(t DIV 5)
B858  250       z=48;         (ORD('0'))
B858  260       lim1=32;      (max. kitevo)
B858  270       lim2=-32;     (min. kitevo)
B858  280       hossz=8;
B858  290 TYPE  posint=0..32;
B858  300 VAR ch,NULL:CHAR;
B858  310     a,y:REAL;
B858  320     i,j,e:INTEGER;
B858  330     q,s,ss:BOOLEAN;
B858  340     ST:ARRAY[1..20]OF CHAR;
B858  350
B858  360 PROCEDURE GET(VAR C:CHAR);
B858  370 BEGIN
B873  380   j:=j+1;
B890  390   IF j<=20 THEN C:=ST[j];
B8E1  400   IF (ORD(C)=8)AND(j>0)THEN j:=j-1;
B933  410   IF j=0 THEN C:='A'
B959  420 END;(GET)
B961  430
B961  440 FUNCTION ten(e:posint):REAL;(=10^e. 0<e<32)
B964  450 VAR i:INTEGER;
B964  460     t:REAL;
B964  470 BEGIN
B97C  480   i:=0;
B985  490   t:=1;
B997  500   REPEAT
B997  510     IF ODD(e) THEN
B9A7  520       CASE i OF
B9AD  530         0:t:=t*1E1;
B9DD  540         1:t:=t*1E2;
BA0D  550         2:t:=t*1E4;
BA3D  560         3:t:=t*1E8;
BA6D  570         4:t:=t*1E16;
BA9D  580         5:t:=t*1E32
BABD  590       END;
BACA  600     e:=e DIV 2;
BADE  610     i:=i+1
BAE9  620   UNTIL e=0;
BB00  630   ten:=t
BB00  640 END;(ten)
BB24  650
BB24  660 PROCEDURE SIGN(VAR a:BOOLEAN);
BB27  670 BEGIN
```

```
BB3F  680   GET(ch);
BB52  690   IF ch='-'THEN
BB6A  700   BEGIN
BB6A  710     a:=TRUE;
BB75  720     GET(ch)
BB7F  730   END ELSE
BB8B  740   BEGIN
BB8B  750     a:=FALSE;
BB95  760     IF ch='+' THEN GET(ch)
BBB7  770   END
BBC0  780 END;(SIGN)
BBC7  790
BBC7  800 BEGIN
BBDF  810   ch:=' ';
BBE4  820   REPEAT
BBE4  830     j:=0;
BBF0  840     AT(Sor,Oszlop+hossz);
BC10  850     FOR i:=1 TO hossz DO
BC33  860     WRITE(CHR(8));
BC40  870     q:=TRUE;
BC45  880     READLN;
BC4B  890     READ(ST);
BC54  900     SIGN(s);
BC61  910     IF NOT (ch IN['0'..'9'])THEN
BC96  920     BEGIN
BC96  930       q:=FALSE;
BC9A  940       GOTO 13
BC9D  950     END;
BC9D  960     a:=0;
BCAF  970     e:=0;
BCB8  980     REPEAT(Egeszresz beolvasasa)
BCB8  990       IF a<limit THEN a:=10*a+ORD(ch)-z
BD0D 1000       ELSE e:=e+1;
BD2E 1010       GET(ch)
BD36 1020     UNTIL NOT(ch IN['0'..'9']);
BD70 1030     IF ch='.'THEN
BD81 1040     (Tortresz beolvasasa)
BD81 1050     BEGIN
BD81 1060       GET(ch);
BD8E 1070       WHILE ch IN['0'..'9']DO
BDC4 1080       BEGIN
BDC4 1090         IF a<limit THEN
BDE7 1100         BEGIN
BDE7 1110           a:=10*a+ORD(ch)-z;
BE27 1120           e:=e-1
BE32 1130         END;
BE34 1140         GET(ch)
BE3C 1150       END
BE41 1160     END;
BE44 1170     IF (ch='e')OR(ch='E')THEN
BE65 1180     (Kitevo olvasas)
BE65 1190     BEGIN
BE65 1200       i:=0;
BE6E 1210       SIGN(ss);
BE7B 1220       IF ch IN['0'..'9']THEN
BEAE 1230       BEGIN
BEAE 1240         i:=ORD(ch)-z;
BEC3 1250         GET(ch);
BED0 1260         WHILE ch IN['0'..'9']DO
BF06 1270         BEGIN
BF06 1280           IF i<limit THEN i:=10*i+ORD(ch)-z;
BF5A 1290           GET(ch)
BF62 1300         END
BF67 1310       END ELSE
BF6D 1320       BEGIN
BF6D 1330         q:=FALSE;
```

```
BF71 1340        GOTO 13
BF74 1350      END;
BF74 1360      IF ss THEN e:=e-1
BF82 1370        ELSE e:=e+1
BFA0 1380    END;
BFB3 1390    IF e<1!m2 THEN
BFCA 1400    BEGIN
BFCA 1410      a:=0;
BFDC 1420      e:=0
BFE8 1430    END ELSE
BFEB 1440    IF e>1!m1 THEN
BFFE 1450    BEGIN
BFFE 1460      q:=FALSE;
C002 1470      GOTO 13
C005 1480    END;
C005 1490    (0<a<2^23)
C005 1500    y:=a;
C01D 1510    IF a THEN y:=-y;
C044 1520    IF e<0 THEN y:=y/ten(-e)
C075 1530    ELSE IF e<>0 THEN y:=y*ten(e);
C0D0 1540 13:UNTIL q=TRUE;
C0E1 1550    ReadReal:=y;
C0F9 1560    AT(Sor,Oszlop+hossz);
C119 1570    FOR j:=0 TO hossz-j DO
C14F 1580    WRITE(CHR(8))
C159 1590 END;(ReadReal)
```

```
C16C 1600
C16C 1610 FUNCTION ReadInteger(Sor,Oszlop:INTEGER):INTEGER;
C16F 1620 (INTEGER szam beolvasasa Sor, Oszlop karakter poziciora)
C16F 1630 VAR a:REAL;
C16F 1640 BEGIN
C187 1650   WHILE (ABS(a)>MAXINT) DO
C1B7 1660   a:=ReadReal(Sor,Oszlop);
C1E1 1670   WHILE NOT (a=TRUNC(a)) DO
C218 1680   a:=ReadReal(Sor,Oszlop);
C242 1690   ReadInteger:=TRUNC(a)
C250 1700 END;(ReadInteger)
C263 1710
C263 1720 BEGIN
C26C 1730   PAGE;
C271 1740   AT(0,0);
C27E 1750   WRITELN('Integer   Real      Real');
C2A4 1760   WRITELN;
C2A7 1770   FOR I:=0 TO 10 DO
C2C1 1780   BEGIN
C2C4 1790     B:=ReadInteger(I+2,0);
C2D8 1800     FOR J:=0 TO 1 DO
C2F2 1810       A:=ReadReal(I+2,J*10+10)
C310 1820   END
C321 1830 END.
End Address: C326
Run?C
```

használhassuk. Ezzel a rutinnal definiáltuk az AT eljárást, melynek segítségével a nyomtatópozícióit a képernyő tetszőleges karakterhelyére állíthatjuk. Az AT hívása a ReadReal függvényből teszi lehetővé azt, hogy a beolvasást a képernyő bármely részén hajtsuk végre.

A ReadInteger függvény, mely egész számok ellenőrzés alatti beolvasására szolgál, ReadReal függvény segítségével egy valós számot olvas be, majd megvizsgálja, hogy ez megfelel-e a Pascal INTEGER számoknak, és ha igen, az eredményt áttölti egy egész számba. Első lépésben azt vizsgálja, hogy az ideiglenesen beolvasott *a* valós szám abszolút értéke MAXINT-nél kisebb-e, majd megvizsgálja, hogy *a* valóban egész szám-e.

A közölt algoritmus elég hosszadalmasnak tűnhet, de a valóságban igen gyorsan működik. Igazság szerint az sem lenne nagy baj, ha lustább volna, hiszen ezt a programrészt interaktív beavatkozásokhoz használjuk, itt pedig az ember mindig sokkal lassabban dolgozik, mint a számítógép.

Megjegyezzük még, hogy a program természetesen csak szintaktikai hibákat szűr ki a beolvasott számokból. Egyéb hibák csak a számítási feladat részletes ismeretében deríthetők fel, és ez csak a felhasználói program készítőjének feladata lehet.

DR. KABOLDY PÉTER

Robot amatőrpályázat

A Magyar Robottechnikai Társaság (MRtT) pályázatot hirdet az ország valamennyi közép- és felsőfokú tanintézményének tanulói/hallgatói, valamint nem felsőfokú végzettségű fiatal műszakiak részére a robottechnika népszerűsítése, a robottechnikai ismeretek és mechatronikai kultúra terjesztése érdekében. Pályázni egyénileg és csoportosan is lehet, részletes felvilágosítást az MRtT ad.

Tárgy: Olyan forgó és/vagy elmozduló csuklókat tartalmazó szerkezet (robot) tervezése és megépítése, amely kisméretű tárgyak megfogására alkalmas, és ezeket adott térrész tetszőleges pontjába tudja vinni, illetve tetszőleges pálya mentén képes mozgatni. A robotnak az alábbi feladatokat kell teljesítenie:

1. **Kötelezően:** adott középpontú és sugarú körök rajzolása, továbbá golyó beejtése adott középpontú síkbeli furatba.

2. **Szabadon választhatóan:** bármilyen, a megalkotott szerkezet teljesítményét, mozgását demonstráló feladat.

A robot specifikációs adatai: mozgási tartomány közelítőleg 50 cm élhosszúságú kocka vagy hasonló sugarú gömb; a mozgatott tömeg max. 100 g; megkívánt működési sebesség: 0,01—0,1 m/s; ismétlési pontosság: +/— 1 mm; a megfogó megkívánt max. nyitása 50 mm.

A pályázat lebonyolítása

I. forduló: Pályázati terv benyújtása *lezárt jeligés borítékban:* a pályázó(k) neve, címe, a felépítés és a működés szabatos leírása, mechanikai rajzok (vonalas szerkezeti vázlat/nézeti rajz/fantázia rajz), a rendszerműködés blokkvázlata, a programrendszer ismertetése, anyagjegyzék, árbecslés. Az anyag A4 formátumú és kezelhető legyen.

II. forduló: A zsűri által realizálhatónak ítélt berendezések elkészítése és bemutatása az alábbi formában: robot (számítógép nélkül) dobozban, működési és kezelési leírás (A4 formátumú lapokon), működtető program.

Értékelési rendszer

Az *I. fordulóban az elbírálás kritériumai:* ötlet, megvalósíthatóság, várható költségek.

A *II. forduló* széles nyilvánosság előtt zajlik — kiállítással egybekötve.

Az értékelés szempontjai: a kitűzött feladat megoldása, a specifikációs követelmények (működési sebesség, pontosság) teljesítése, a szabad feladat ötletessége, kezelhetőség, külső megjelenési forma, kivitel, ár.

Díjazás: A második fordulóba bejutott pályázók a berendezés megépítésének támogatása mellett 2—2000 Ft díjazásban részesülnek.

Pályadíjak: 1. díj: 20 000 Ft (1 db), 2. díj: 10 000 Ft (2 db), 3. díj: 5000 Ft (4 db). MM különdíj a legjobb középiskolai pályázónak: 10 000 Ft.

A helyezettek lehetőséget kapnak arra, hogy berendezésükről cikket jelentessenek meg. Ha a pályázat során szabadalommal védhető eredmény születik, a szerző a megfelelő jogvédelemben részesül.

Tanácsadás, segítség, felvilágosítás

Konzultációs lehetőségek az **I. fordulóban:** Bánki Donát Gépipari Műszaki Főiskola: dr. Kégl Tibor (338-967), Budapesti Műszaki Egyetem: Dr. Filemon Józsefné (664-011/13-67), Dr. Lantos Béla (664-011/20-58), Eötvös Loránd Tudományegyetem: dr. Déry József (188-643), Gépipari és Automatizálási Műszaki Főiskola: dr. Kulcsár Béla (76-20-567), Kandó Kálmán Villamosipari Műszaki Főiskola: dr. Kelemen Ferenc (804-511), Nehézipari Műszaki Egyetem: dr. Szintai István (46-66-111), MTA—SZTAKI—FLEXYS: Borsay György (552-404).

A **II. fordulóhoz** az MRtT felkéri az érintett gyárakat, intézeteket a pályázat támogatására alkatrészekkel, műhelykapacitással, konzultációs, technológiai és programozási tanácsadással.

Határidők: Az I. forduló pályaműveinek **beadási határideje:** 1987. 09. 31. *Eredményhirdetés:* 1987. 12. 31. körül. A II. forduló meghirdetésének napja: 1988. 01. 01-ig. A II. forduló *beadási határideje:* 1988. 06. 31. *Eredményhirdetés:* 1988. 12. 31-ig.

A pályázatok benyújtási helye: MTA—SZTAKI—FLEXYS, dr. Hajnal Miklós, Budapest, Bíró u. 9/b. 1122

Ipari park a műegyetem mellett

A görögországi számítástechnika helyzete azért is érdekes Magyarországon, mert egy hazánkkal jól összemérhető országról van szó: területe 132 ezer km², lakossága 9,5 millió. Még a főváros-központúság is ismétlődik: Athén több mint kétmilliós nagyváros, amely koncentrálja az ipar és a szellemi élet javát.

Sétáltunk az athéni belvásorban, és meglepetéssel tapasztaltuk, hogy csak elvétve láthatók a kirakatokban mikroszámítógépek. Ez furcsa ellentétben állt az országról addig alkotott összképpel, ezért az egyik görög kollégához fordultunk. Nevetve válaszolt: „Athénben minden árucikket az erre szakosított körzetekben kínálnak. Így találtok olyan városrészt, ahol minden üzlet például kerékpárokat vagy fürdőszoba-berendezéseket árusít. Ugyanígy van ez a számítógépek esetében is. Látogassatok csak el a Stournar utcába, a Műszaki Egyetem mellé!

A kellemes, fákkal szegélyezett utácska egyik oldalán a műegyetem épülettömbje, parkja, a másik oldalán üzletsor. Valóban, egymás mellett sorakoznak a számítástechnikai szaküzletek, intézmények. Jelentős hányaduk foglalkozik hardver-szoftver értékesítéssel, szakkönyvek árusításával, és ezek túlnyomórészt a mikroszámítógép-gyártó cégek szerint szakosodtak; így találtunk Commodore, Amstrad, Philips szaküzletet. A szoftver-, illetve a szakkönyvválaszték természetesen az árusított hardver függvénye. Ezekben az üzletekben a mélyebb ismereteket adó szakkönyveket kínálták, például a Lotus 1—2—3-ról négy-ötfélét is, angol nyelven. Görög nyelvű irodalom csak főképpen a programnyelvekhez, valamint a nagyobb tömegben elterjedt géptípusokhoz volt: láttunk görög nyelvű Spectrum- és Commodore-könyveket. Ez utóbbiakat a város más részein lévő könyvesboltokban is árusították, de ott speciálisabb számítástechnikai szakkönyveket, például dBASE-leírást keresni már reménytelen volt.

A Stournar utca számítástechnikai üzleteinek másik részét tanácsadással, szoftverfejlesztéssel foglalkozó kis cégek foglalják el. Ők is forgalmaznak kész gyári szoftvereket és szakkönyveket, bár ezek szerepe szemmel láthatóan inkább csak a becsalogatásra korlátozódik. Ezek a kis cégek konkrét témákra szakosodnak, és szívesen eligazítják azokat, akik valamilyen problémával hozzájuk fordulnak, akár a pár sarokkal odébb lévő társcéghez is átkísérik az érdeklődőt.

Az utcában több klub is működik, méghozzá a házak felsőbb emeletein. Egy kivételt találtunk csupán: hatalmas Commodore-emblémákra, nagy klubfeliratokra lettünk figyelmesek az egyik földszinti üzlethelyiségen. Hamarosan kiderült, hogy ez eredetileg étterem, amely beállított néhány Commodore gépet, hogy odavonzza s lehetőleg sokáig ott is tartsa az ifjú kuncsaftokat. Az evés-ivás mellett itt akár programcserére is lehetőség nyílott. Ez már több, mint okos vendéglátás — alkalmazkodás a környezethez —; ez már szinte azonosulás.

Rendkívül tanulságos volt ez a séta; érzékletesen bizonyította a műhely, azaz a műegyetem jelentőségét, szellemi kisugárzásának valóságos határait, táguló terét. Szemünk előtt a Budapesti Műszaki Egyetem mellé tervezett ipari park látomása lebegett, mintha a jövőben jártunk volna. Reméljük, a nem túl távoliban . . .

DR. BROCZKÓ PÉTER

Jellemző mikroszámítógép-árak

Típus	Ár	
	(ezer drachma)	(ezer forint)
ZX—Spectrum	20—22	6—7
ZX—Spectrum +	22—28	7—9
ZX—Spectrum 128 k	33	11
ZX—Spectrum QL	35	12
Commodore 64	54	18
Commodore Plus 4	30	10
Commodore 128	67	22
Commodore 128D	149	50
Philips MSX	45	15
Sanyo 64 k MSX	30	10
Atari 520 ST	170	57
Atari 1040 ST	240	80
Spectravideo 328	46	15
Amstrad 128	95—120	32—40
Amstrad 464	90	30
Amstrad 512	145	48
Dragon 32	27	9
Dragon 64	40	13
Commodore 801 nyomtató	40	13
1 db 3,5 hüvelykes lemez	1,5	0,5

Számítógépüzlet a Stournar utcában *(A szerző felvétele)*

Vigyázat! Tolvaj!

Eddig már háromfajta hibagenerálással ismerkedtünk meg. Elsőként a 21-es típusú hibát tárgyaltuk, amely akkor lép fel, ha az FDC (Floppy Disk Controller) nem talál szinkronkaraktert. Ezután láttuk, hogyan lehet olyan lemezhibát (20-as hiba) előállítani, amelynek hatására a VC—1541 nem találja a blokk fejlécét; majd a 22-es hibával foglalkoztunk, amely akkor lép fel, ha az FDC nem találja az adatblokkot.

A sort a 29-es lemezhibával folytatjuk. Ezt a hibát az okozza, hogy a lemez különböző blokkjaiban más-más a lemezazonosító, az ID.

Az FDC minden egyes blokk elolvasása előtt megvizsgálja, hogy a blokk azonosítója megegyezik-e az adott lemez 18. sáv 0. blokkjának azonosítójával, ahol a lemezkatalógus kezdődik. Itt található a 140 bájtnyi BAM (Block Availability Map), amely a blokkok foglaltságát mutatja. Ugyanitt találjuk a lemez nevét és a lemezformattálásnál adott azonosítót. Erről a lemezmonitorról könnyen meggyőződhetünk.

A szakirodalomban gyakran felhívják a figyelmet arra, hogy az itt lévő ID-t a formattálás után, monitorból ne írjuk át. Ezt azzal indokolják, hogy ha több egyforma azonosítójú lemezt használunk közvetlenül egymás után, akkor előfordulhat, hogy az egyik BAM-ja a másik lemezre íródik, és így azon az adatok hozzáférhetetlenné válnak, esetleg megsemmisülnek. Ez azonban így nem igaz. A monitorral írható és olvasható ID-szám ugyanis csak tájékoztató jellegű; az FDC a blokk fejrészében lévő ID szerint azonosít. A 18. sáv 0. szektorának blokkfejében talált ID-számot eltárolja a RAM-ba, és olvasáskor ezzel hasonlítja össze a többi blokk azonosítóját.

Az elmondottakból következik, hogy ha a lemez bizonyos sávjait a katalógus sávjától eltérő azonosítóval formattáljuk, akkor a lemezt védetté tesszük, hiszen ezt követően a lemez már csak egy segédprogram segítségével írható és olvasható. A segédprogram a lemez inicializálása után átírja a meghajtó memóriájában az ID-t arra a számértékre, amellyel a kritikus blokkokat formattáltuk; ezután el tudjuk olvasni azokat. A RAM-ban az ID visszaállítása inicializálással egyszerűen megoldható.

A DOS INSIDE alapján készült az első példaprogram, a második pedig a lemez sávjait más-más ID-vel formattálja.

KOVÁCS P. ATTILA

```
100 REM FORMAT A DISKETTE - 1541
110 DIMT(35),H$(35),L$(35)
120 PRINT"(CLR)FORMAT A DISKETTE - 1541"

130 PRINT"(DOWN)INSERT (RVS)MASTER(ROFF)
 IN DRIVE"
140 GOSUB910
150 PRINT"(DOWN)(RVS)FETCHING(ROFF) FORM
ATTING ID"
160 OPEN15,8,15
170 FORI=1TO35
180 T(I)=1
190 NEXTI
200 JOB=176
210 FORT=1TO35
220 IFT(T)=0GOTO340
230 GOSUB970
240 IFE=1GOTO280
250 H$(T)=CHR$(0)
260 L$(T)=CHR$(0)
270 GOTO340
280 PRINT#15,"M-R"CHR$(22)CHR$(0)
290 GET#15,H$(T)
300 IFH$(T)=""THENH$(T)=CHR$(0)
310 PRINT#15,"M-R"CHR$(23)CHR$(0)
320 GET#15,L$(T)
330 IFL$(T)=""THENL$(T)=CHR$(0)
340 NEXTT
350 T=18
360 GOSUB970
370 CLOSE15
380 PRINT"(CLR)FORMAT A DISKETTE - 1541"

390 PRINT"(DOWN)INSERT (RVS)BLANK(ROFF)
 IN DRIVE"
400 GOSUB910
410 OPEN15,8,15
420 FORJ=0TO6
430 FORI=0TO7
440 READD
450 D$(J)=D$(J)+CHR$(D)
460 NEXTI
470 NEXTJ
480 I=0
490 FORJ=0TO6
500 PRINT#15,"M-W"CHR$(I)CHR$(4)CHR$(8)D
$(J)
510 I=I+8
520 NEXTJ
530 FORI=1TO35
540 PRINT#15,"M-W"CHR$(49+I)CHR$(4)CHR$(
1)L$(I)
550 PRINT#15,"M-W"CHR$(84+I)CHR$(4)CHR$(
1)H$(I)
560 NEXTI
570 REM EXECUTE
580 PRINT"(DOWN)(RVS)FORMATTING(ROFF) DI
SKETTE"
590 PRINT#15,"M-E"CHR$(0)CHR$(4)
600 INPUT#15,EN$,EM$,ET$,ES$
610 T=18
620 S=0
630 JOB=176
640 GOSUB970
650 JOB=128
660 GOSUB970
670 PRINT#15,"M-W"CHR$(0)CHR$(4)CHR$(3)C
HR$(18)CHR$(1)CHR$(65)
680 JOB=144
690 GOSUB970
700 S=1
710 JOB=128
720 GOSUB970
730 PRINT#15,"M-W"CHR$(0)CHR$(4)CHR$(2)C
HR$(0)CHR$(255)
740 JOB=144
750 GOSUB970
760 CLOSE15
770 OPEN15,8,15
780 PRINT#15,"NO:1541 FORMAT"
790 INPUT#15,EN$,EM$,ET$,ES$
800 S=0
810 JOB=128
820 GOSUB970
830 PRINT#15,"M-W"CHR$(162)CHR$(4)CHR$(2
)CHR$(50)CHR$(54)
840 JOB=144
850 GOSUB970
860 PRINT#15,"M-W"CHR$(162)CHR$(7)CHR$(2
)CHR$(50)CHR$(54)
870 CLOSE15
880 PRINT"(DOWN)DONE!"
890 END
900 REM DELAY
910 PRINT"(DOWN)PRESS (RVS)RETURN(ROFF)
TO CONTINUE"
920 GETC$:IFC$=""THEN920
930 IFC$<>CHR$(13)GOTO920
940 PRINT"OK"
950 RETURN
960 REM JOB QUEUE
970 TRY=0
980 PRINT#15,"M-W"CHR$(8)CHR$(0)CHR$(2)C
HR$(T)CHR$(S)
990 PRINT#15,"M-W"CHR$(1)CHR$(0)CHR$(1)C
HR$(JOB)
1000 TRY=TRY+1
1010 PRINT#15,"M-R"CHR$(1)CHR$(0)
1020 GET#15,E$
1030 IFE$=""THENE$=CHR$(0)
1040 E=ASC(E$)
1050 IFTRY=500GOTO1070
1060 IFE>127GOTO1000
1070 RETURN
1080 REM NEW
1090 DATA169,  0,133,127, 32,  0,193,169

1100 DATA 76,141,  0,  6,169,199,141,  1

1110 DATA  6,169,250,141,  2,  6,169,224

1120 DATA133,  3,164, 81,185, 49,  4,133

1130 DATA 19,185, 84,  4,133, 18,192, 35

1140 DATA208,240,165,  3, 48,252, 76,148

1150 DATA193,234,234,234,234,234,234,234
```

MULTIPLE ID FORMATTING SOURCE LISTING

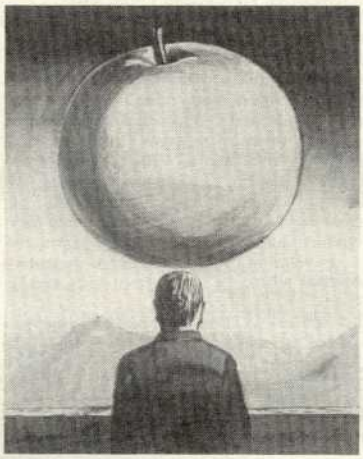
```
100 REM FAD.PAL
110 REM
120 OPEN2,8,2,"@0:FAD.B,P,W"
130 REM
140 SYS40960
150 ;
160 .OPT P,O2
170 ;
180 *= $0400
190 IDL = $0431
200 IDH = IDL+35
210 ;
220 LDA #$00
230 STA $7F         ; DRIVE NUMBER
240 ;
250 JSR $C100       ; LED
260 ;
270 LDA #$4C        ; JUMP TO $FAC7
280 STA $0600
290 LDA #$C7
300 STA $0601
310 LDA #$FA
320 STA $0602
330 ;
340 LDA #$E0
350 STA $03
360 ;
370 TABLE LDY $51 ; TRACK NUMBER
380 ;
390 LDA IDL,Y       ; ID LO
400 STA $13
410 ;
420 LDA IDH,Y       ; ID HI
430 STA $12
440 ;
450 CPY #$23        ; TRACK 35
460 BNE TABLE
470 ;
480 WAIT LDA $03
490 BMI WAIT
500 ;
510 JMP $C194
```

Mesterséges értelem III.

Reprezentációk

Képzeljük magunkat egy percre Pamacs kutya helyébe. A szeretett gazdi éppen elhajította kedvenc botunkat. Mi persze lelkesen rohanunk, hogy visszahozzuk. Ám egyszer csak kerítésbe ütközünk: jobbra is kerítés, balra is kerítés — nem tudunk a bothoz közelebb kerülni. Ha viszont a gazdi szemével nézzük a kerítésen átdobott botot, akkor azonnal végiggondoljuk, hogy ugyan a fizikai távolság egy darabig növekedni fog, ha az odébb levő kertkapun megyünk át, de ekkor valamilyen értelemben mégis közelebb kerültünk a bot visszaszerzéséhez.

Látható, hogy ez esetben a probléma leírása, jellemzése a fizikai távolsággal csak a kerítés nélküli esetekben megfelelő, hiszen csak ekkor visz a fizikai távolság csökkenése egyértelműen közelebb a célhoz, illetve növelése távolabb tőle. Olyan leírásra, mértékre lenne mindig szükség, amely tekintetbe veszi az összes lehetséges műveleteket, és amelyben ugyanakkor a kiindulási és a célállapot távolsága a legkisebb, azaz a végrehajtható műveletekkel a legkönnyebben áthidalható. Ráadásul nemcsak ilyen egyszerű helyzeteket kell jól leírni, hanem egyéb, a világra vonatkozó ismereteket is, amelyek más problémák, például egy mondat, egy látvány értelmezéséhez vagy egy matematikai feladat megoldásához tartozó ismereteket hordozzák magukban. A világot tükröző ismeretfajtákhoz más és más leírási, reprezentálási módszer látszik alkalmasnak. Hogyan találhatjuk meg minden feladathoz a legmegfelelőbb leírást? Ez ma a kutatások igen fontos kérdése. Jelenleg még nincs a reprezentációknak általános elméletük, amely egységes keretbe foglalná a különböző leírási módokat, indokolná, hogy melyik milyen esetben ajánlott.



1. kép. A levelezőlap

Az ismeretek fajtái

Az intelligens viselkedéshez szükséges tudást típusokra szokták osztani. Ezek megkülönböztethetősége és egyáltalán fajtái érdekes filozófiai, pszichológiai kérdések. A típusok közötti határok általában nem élesek, hiszen a különböző ismeretfajták többé-kevésbé átválthatók egymásba. Először tekintsük át, hogy milyen ismeretfajtákat különböztetnek meg.

Tárgyi tudás, illetve tények. „A hó fehér." „A madaraknak szárnyaik vannak." Természetesen ide tartozik a tárgyak és fogalmak osztályozása, valamint tulajdonságaik leírása is.

Eseményekre vonatkozó ismeretek. „Holnap leszakad az ég." *(1. kép)* Itt az események időbeli sorrendjét és az oksági kapcsolatokat is számon kell tartani.

Eljárások. Hogyan kell biciklizni vagy Pascalban programozni. A tárgyi és eljárási tudás közötti határ különösen nehezen vonható meg. Tulajdonképpen az eljárási tudás részben kifejezhető tárgyi ismeretekkel is, de ennek erőltetése alapvető elvi problémákhoz vezet.

Metaismeretek. Arra vonatkozó ismeretek, hogy mit tudunk. Egy tárgyra vonatkozó ismereteink kiterjedésének mértéke, az ismereteink forrása és megbízhatóságuk, továbbá egyes ismeretek fontosságának viszonya. Tegyük fel, hogy a város egy másik pontjára akarunk átjutni. Választhatunk különböző stratégiák között: találomra kérdezősködni kezdünk vagy térképet viszünk. Ha az utóbbi mellett döntünk, akkor máris felhasználtuk az olvasási képességünkre vonatkozó ismeretünket — azaz, hogy mivel tudunk olvasni, ezért utunk során majd el is olvassuk az utcanévtáblákat és a térképet, és ily módon képesek leszünk menet közben is tájékozódni.

Az ismeretek felhasználása

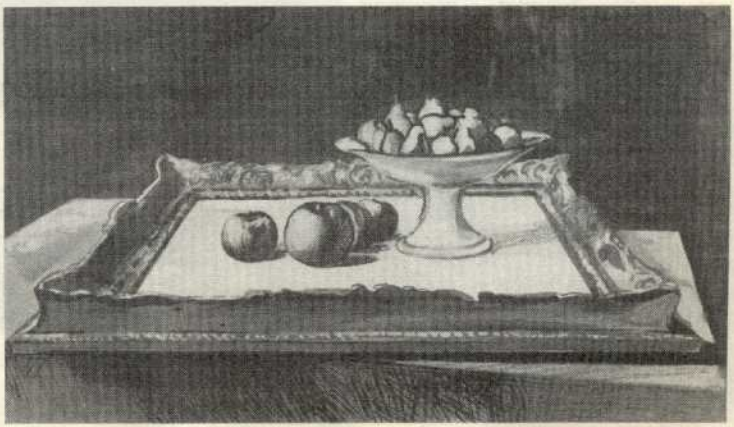
A mesterséges intelligencia programokban az ismeretek kezelése, felhasználása három szakaszra bontható; ezek azonban sokszor egymásba fonódnak.

Ismeretek gyűjtése. Tanulási céllal az ismeretek gyűjtése több, mint az újak egyszerű hozzáadása a régebbiekhez. Ennek megfelelően az MI-rendszerek — mielőtt beiktatják az adatbázisba — gyakran osztályozzák, feldolgozzák az új ismereteket, hogy a későbbiek során könnyebb legyen megtalálni az éppen aktuális problémához tartozókat. Sokszor, amikor a régi és új ismeretek összeépülnek, az addig hibátlanul végzett feladatokban is átmenetileg zavart okozhatnak. Ha viszont az új ismeretek nem épülnek be szervesen a régiek közé, akkor a viselkedés általános intelligenciája szinte egyáltalán nem fokozódik.

Visszakeresés. Mivel rengeteg ismeret áll rendelkezésre, ezért mindig kritikus az adott probléma megoldásához lényeges ismeretek megtalálása. Az emberek ebben rendkívül ügyesek. A programok számára az a legfontosabb, hogy az ismeretek gyűjtése során lehetőleg megtörténjen az információk kapcsolása és csoportokba gyűjtése.

Következtetés. Amikor a rendszernek olyan dolgot kell csinálnia, amiről nem rendelkezik valamennyi explicit ismerettel, akkor következtetnie kell abból, amit már tud *(2. kép)*. Minden egyes reprezentációs

2. kép. A józan ész



sémával kapcsolatban nagyon lényeges kérdés, hogy egyáltalán milyen jellegű következtetési módok érvényesülhetnek, és ezen belül melyek egyszerűek és természetesek az adott formalizmusban. Általában elfogadottak az alábbiak:

— Formális következtetés az adatok szintaktikus manipulálása az előzetesen definiált következtetési szabályok szerint. Ennek alkalmazásával épül fel a matematikai logika.

— Eljárás jellegű következtetés során szimuláció segítségével keressük a választ kérdésekre, vagy megoldást a problémákra; lejátsszuk a különböző eseménysorokat, amelyeket cselekedeteink eredményezhetnek.

— Az analógiás következtetés az ember számára legtermészetesebbnek tűnő, de programban nagyon nehezen megfogalmazható módszer. Például: „Tudom, hogy a rigók tudnak repülni, a verebek lényegében olyanok, mint a rigók, akkor valószínű, hogy tudnak repülni." A programban való megvalósítás nehézségét a „lényegében" szó okozza. Vagyis annak absztrakt megfogalmazása, hogy mely esetekben áll fenn a lényegi hasonlóság két dolog között.

— Az általánosítás és elvonatkoztatás az előzőhöz hasonlóan az ember számára szintén természetes, de nehezen programozható. Például: „Ha tudom, hogy a galamboknak van szárnyuk, a rigóknak is van szárnyuk, és a verebeknek is van szárnyuk, akkor arra következtetek, hogy minden madárnak van szárnya."

— A metakövetkeztetés az ismeretek meglétére és minőségére vonatkozó információk felhasználásával történő következtetés. Az újabb kutatások szerint ez a fajta következtetés központi szerepet játszik az emberi gondolkodásban.

A reprezentációs sémák jellemzői

A mesterséges intelligencia kutatások körében ismeretreprezentációs sémán egy-egy speciális adatstruktúrát és a hozzá tartozó — az ismeretekből értelmes következtetések levonását lehetővé tevő — interpretáló eljárás együttesét értjük. Az alkalmazott adatstruktúrák vagy valamilyen általános érvényűnek tűnő matematikai formalizmuson alapulnak, mint például a logikai reprezentáció (lásd alább), vagy az emberi adat-, ismeretkezelés megfigyeléséből és utánzásának kísérletéből jöttek létre.

A tárgyalandó reprezentációs sémák mindegyikét alkalmazták már sikeresen a bizonyos mértékig értelmes viselkedést mutató programokban. Ugyanakkor nem tudjuk, hogy miért bizonyulnak egyes sémák adott feladatosztályokra alkalmasabbnak a többinél, valamint az egyes sémák pszichológiai érvényessége is erősen kérdéses *(3. kép)*. Bizonyos dolgok bizonyos reprezentációkban „könnyebben" kifejezhetők, de nincs formális mértéke annak, hogy egy reprezentáció egy adott esetben mennyire jó. Egyáltalán, nincs számszerű összehasonlítási lehetőség a reprezentációs sémák között.

A következőkben a reprezentációs sémákkal szemben támasztott követelmények közül sorolunk fel néhányat. E követelmények általában ellentmondóak, és a döntés nehéz, hiszen a kompromisszumot mindig a feladatnak megfelelően kell megtalálni.

Hatókör és részletesség

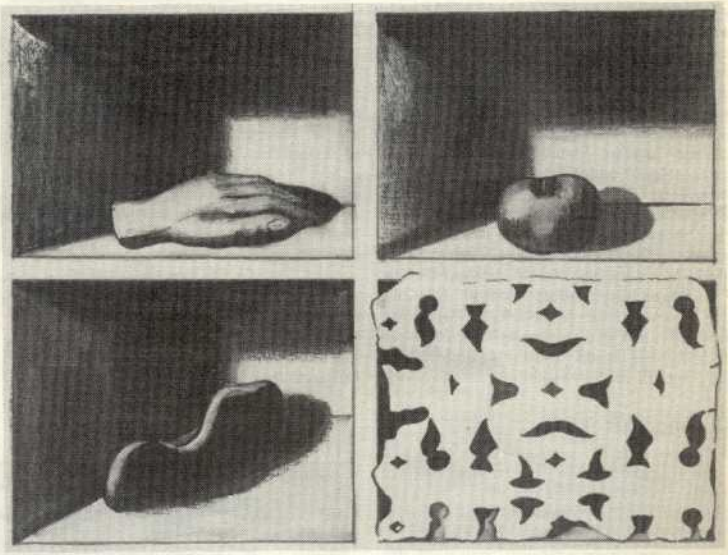
A hatókör és részletesség fogalomkörén azt értjük, hogy a külső világ mekkora szelete és milyen felbontással, pontossággal reprezentálható az adott sémában. Túl részletesen egy szélesebb terület már kezelhetetlen, ugyanakkor egy konkrét megoldáshoz minden szükséges ismeretnek rendelkezésre kell állnia. Azt, hogy milyen részletesség elégséges a következtető mechanizmus számára, a rendszer tervezése során kell mérlegelni.

Határozatlanság és a szemantikus alapelemek

Bármely reprezentációs formalizmusban a rendszer információs, szemantikus egységkészletének kiválasztása, valamint az egyes elemek által kifejezett információ összetettségének mértéke döntő a séma kifejezőkészségét illetően. A szemantikus alapelemek kiválasztása befolyásolja azt, hogy egy bizonyos ismeret hány különböző módon írható le az alapelemek segítségével *(1. ábra)*.

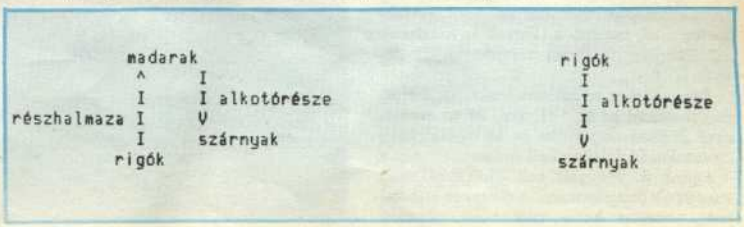
Az ábrán látható, hogy az első leírás határozatlanabb, mint a második, ugyanak-

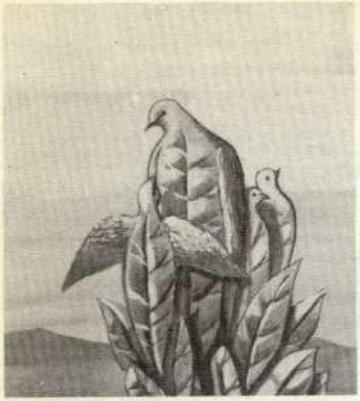
3. kép. Az est jelei



4. kép. Egy éjszaka múzeuma

1. ábra. „A rigóknak szárnyuk van" állítás kétféle leírási módja





5. kép. Természetes bájak

kor több információt is tartalmaz.[1] Hiszen nemcsak azt állítja, hogy a rigóknak van szárnyuk, hanem egyben azt is, hogy a rigók madarak, és minden madárnak van szárnya. Ez az esetenként rendkívül hasznos határozatlanság azonban gyakran okoz zavart, mert a rendszer nem tudja, hova forduljon, amikor egy adott információt vissza akar keresni.

Modularitás és érthetőség

A modularitás foka azt fejezi ki, hogy a rendszer egyes elemei mennyire függetlenek egymástól és a rendszer többi részétől; vagyis hogy mennyire könnyű új elemeket hozzáadni, valamint a már meglévőket módosítani, illetve törölni *(4. kép)*. Ha egy rendszerben az ismereteket többségében tényszerű állítások és nem eljárások tartalmazzák, vagyis a rendszer deklaratív megközelítésű, akkor általában modulárisabb. Ez abból adódik, hogy minél közvetlenebbül vannak kifejezve a rendszert alkotó ismeretek, annál áttekinthetőbbek kapcsolataik és egymáshoz való viszonyuk az ember számára, és ez egyben egy-egy módosítás következményeinek átláthatóságát is magával hozza. Ha viszont egy rendszerben levő ismeretanyag jórészt eljárásokban testesül meg (procedurális megközelítés), akkor a rendszer tartalma ugyan kevésbé átlátható, de működése könnyebben követhető.

Az ismeretreprezentáció szempontjából viszont az jelent nehézséget, hogy az emberek egyes ismeretfajtái — különösen az eljárás jellegű ismeretek — modulokra bontása rendszerint idegen az emberi természet számára.

A reprezentációs sémák áttekintése

Az itt említendő reprezentációs sémákat azért kell kiemelni a mesterséges intelligencia programokban egyáltalán előforduló rengeteg próbálkozás közül, mert nemcsak egy-egy speciális esetben, hanem számos alkalmazásban beváltak már. Ez a viszonylag általános felhasználhatóság azt igazolja, hogy ezek a sémák valószínűleg sikeresen ragadják meg az emberi gondolkozási mód egyes összetevőit.

Az állapottér explicit definiálása

Az állapottér a probléma struktúráját tulajdonképpen a lehetséges állapotokban megengedett választási lehetőségek, állapotátmenetek felsorolásával írja le (lásd sorozatunk első részét).

Logikai reprezentáció

Ez a klasszikus matematikai megközelítés arra, hogy a világra vonatkozó ismereteinket reprezentáljuk. Például: minden x-re, ha Madár(x)→ Szárnya—van(x) =
= minden objektumra igaz, ha az madár, hogy van szárnya

Ennek a reprezentációnak az előnye, hogy léteznek az ún. logikai következtetési szabályok, amelyek segítségével ismert tényekből további állításokat nyerhetünk, amelyek garantáltan igaziak *(5. kép)*. Például ha tudjuk, hogy a veréb is madár, azaz
minden x-re, ha Veréb(x)→ Madár(x)
akkor ebből következik annak az állításnak az igazsága, hogy a verebeknek szárnyuk van:
minden x-re, ha x Veréb(x)→ Szárnya—van(x)

A következtetések megbízhatósága, konzisztenciája (ellentmondásmentessége) és gépiesen alkalmazható szabályainak bizonyító ereje a logikai reprezentáció óriási előnye a többi reprezentációs sémával szemben.

Eljárás jellegű reprezentációk

Ezekben a sémákban a világra vonatkozó ismeretek eljárásokba vannak belefoglalva, amelyek „tudják", hogy mi a teendő bizonyos jól meghatározott helyzetekben. Ilyen jellegű reprezentáción alapulnak egyes nyelvértelmező rendszerek. Egy mondat azonosítása után a mondatrészek azonosítása következik. Itt az erre szolgáló eljáráshívások reprezentálják, hogy a mondat alanyból, állítmányból stb. áll. Az állítmányban szintén eljáráshívásként szerepel az, hogy összetett is lehet és így tovább. Például egy BASIC-ben írt számítógépprogramban vannak a környezetre vonatkozó ismeretek ilyen módon beágyazva.

Szemantikus hálók

A szemantikus háló, amit Quillian fejlesztett ki 1968-ban, az emberi asszociációs memóriát kívánta utánozni. A háló csomópontokból és összeköttetésekből áll, amelyek a csomópontok kapcsolatát hivatottak jelképezni. A csomópontokban tárgyak, tulajdonságok, illetve fogalmak helyezkednek el, az őket összekötő élek pedig kapcsolattípusokat, viszonyokat jelölnek. A kapcsolatok explicit kifejezése jelentősen megkönnyítheti a keresést egy nagy adatbázisban *(2. ábra)*.

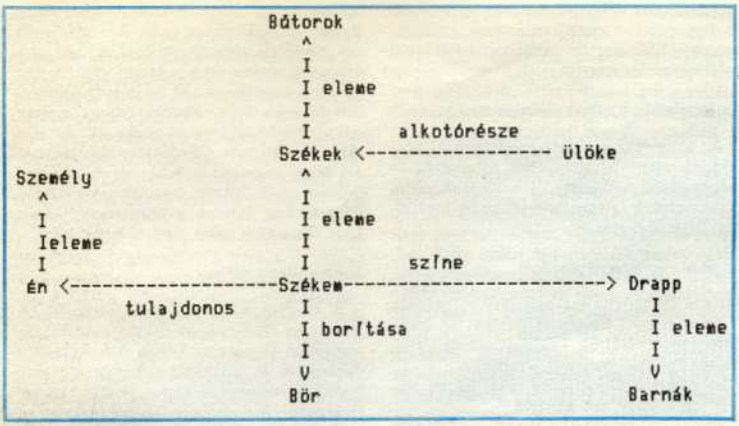
A direkt rámutatás különösen kellemes a tulajdonságöröklődést meghatározó kapcsolatok, a RÉSZHALMAZ és az ELEME esetén *(3. ábra)*.

A tulajdonságöröklődés által minden, az öröklési láncban lejjebb álló dolog örökli a felette állók tulajdonságait *(6. kép)*. Ha tehát például tudjuk, hogy minden szárazföldi állat levegőt lélegez be, akkor biztosak lehetünk abban, hogy a vízben minden rigó megfullad.

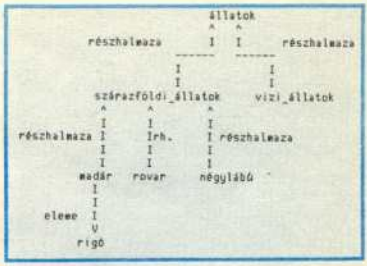
Produkciós rendszerek

A szabályokat tartalmazó produkciós rendszerek előnye, hogy a szabályok feltételrészében egyértelműen és könnyen eldönthetően kódolva van, hogy az adott sza-

6. kép. Kollektív képzelet



2. ábra. A székemmel kapcsolatos néhány ismeret szemantikus hálóba szervezve



3. ábra. Tulajdonságöröklődést bemutató hálózat az ELEME és RÉSZHALMAZ reláció felhasználásával

bály mikor alkalmazható. A produkciós rendszerek esetében elvben az egyes szabályok egymástól mind jobban való elkülönítésére kell törekedni. A szabályok felsorolásából álló reprezentáció előnye az újabb ismeretek automatikus hozzáadásának egyszerűsége, hiszen itt nincs szükség például a mutatók beállítására és egyéb, az új ismeretnek a már meglévőkhöz való integrálását szolgáló külön műveletekre (lásd sorozatunk első részét).

Keretek (frames)

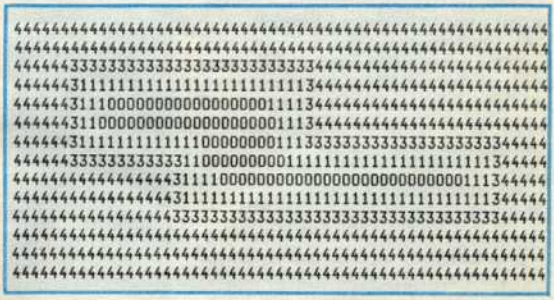
A reprezentációs kutatások legfrissebb eredménye a keretforma: egy olyan adatstruktúra, amely belső viszonyok alapján egyesít deklaratív, illetve eljárás jellegű információt *(7. kép)*. Például egy keret a kutya fogalom számára olyan mezőkből állhat, mint FAJTA, GAZDA, NÉV, továbbá egy a GAZDA mezőhöz rendelt eljárásból, ami meghatározza a tulajdonost, ha az nem ismert *(4. ábra)*.

```
Általános KUTYA keret

      Azonosító : ÁLLAT , JÁTSZÓPAJTÁS
      Fajta     : (...)
      Tulajdonos: SZEMÉLY
                  (ha szükséges : keress egy SZEMÉLYT JÁTSZÓPAJTÁSSAL)
                  (default = ÉN)
      NÉV       : ERVÉNYES NÉV
                  (default = BODRI)

SZOMSZÉD-KUTYA keret

      Azonosító : KUTYA
      Fajta     : NÉMET JUHÁSZ
      Tulajdonos: JÓZSI BÁCSI
      Név       : KORMOS
```

4. ábra. Példa a KUTYA és SZOMSZÉD KUTYA keretek megvalósítására

A keretséma előnye, hogy képes eldönteni saját alkalmazhatóságát. Amennyiben például a KUTYA keret nem talál megfelelő tulajdonost, akkor a keresési eljárás végén meghívhatja a TALÁLT-KUTYA vagy a SZOMSZÉD-KUTYA keretet, és így tovább.

7. kép. A barbár



5. ábra. Két egymást átfedő, kb. azonos fényességű téglalapot ábrázoló kép világosságmátrixa. Az egyes számértékek a nekik megfelelő pont fényességértékeit jelentik

Speciális technikák

A speciális technikákat azért szükséges megemlíteni, mert ugyan nem általánosan alkalmazható sémákat valósítanak meg, de egy speciális feladatban — amelyhez kidolgozták, „finomhangolták" őket — rendkívül eredményesek lehetnek.

Ilyen például a Gelernter által konstruált, geometriai tételeket bizonyító programban alkalmazott módszer. A program bemenete egy geometriai ábra volt, amelyet úgy kellett megválasztani, hogy a lehető legáltalánosabb esetet tartalmazza a véletlen egybeesések kiküszöbölésére. A program ezután az ábrával együtt megadott tételt kísérelte meg bizonyítani. A feladatról

menet közben kialakuló „elképzeléseit" pedig az ábrán ellenőrizte. A program annyira sikeres volt, hogy egy, már az ókor óta ismert geometriai feladatra talált az addig ismerteknél egyszerűbb megoldást.

Hogy érzékelhessük azt, hogy ezek a sémák milyen szűk területen alkalmasak, tekintsük a következő példát. Kódoljuk a robot kamerája által észlelt látványt egy mátrixban, amelynek elemei a megfelelő képpontok világosságértékeit tartalmazzák *(5. ábra)*. Ez a közvetlenül a fizikai tulajdonságokon alapuló leírás bizonyos feladatokra, mint például az objektum széleinek meghatározása, alkalmas, de szinte teljesen alkalmatlan arra a kicsit eltérő feladatra, hogy meghatározzuk a képen látható objektumok számát. Ez két részben átfedő, azonos fényességű objektum esetéből nyilvánvaló. Ezeknél nem dönthető el ennek a leírásnak

8. kép. Honvágy

alapján, hogy egy vagy két objektumról van-e szó, pedig a környezettől elválasztó határvonal könnyedén megállapítható.

* * *

Befejezésképpen szeretnénk újra hangsúlyozni a reprezentációkkal kapcsolatos elméleti nehézségeket *(8. kép)*, amelyeket megoldó összefoglaló elmélet még várat magára. A különböző reprezentációk többkevesebb nehézséggel egymásba átválthatók; így azt, hogy az adott feladatra melyik alkalmasabb, mindig a program belső szempontjai alapján kell eldönteni.

A megnyugtató megoldásra valószínűleg még egy darabig várni kell. Érdekes eredményekre számíthatunk az erősen párhuzamos felépítésű számítógépek vagy a szintén idevágó kutatás, az asszociatív felépítésű memóriák területéről.

SOMOGYVÁRI KÁROLY

(A cikket illusztráló képek RENÉ MAGRITTE után készültek)

Az adattípus fejlődése I.

Hajdanában, fiatal szilvafa koromban én is úgy gondoltam, hogy különböző szilárdságú tudományok vannak. A történelem, a művészettörténet, a pszichológia lágyabb tudományok, amelyek ugyanannak a dolognak több különböző értelmezését is megadhatják, és ezek egyike sem abszolút igaz vagy abszolút elvetendő. A természettudományok keményebbek; itt az igazság általában reprodukálható, míg az ellenkezője cáfolható. És mindenekfelett ott trónol a matematika, az ő véglegesen bebizonyított, helytől és időtől független igazságaival.

Az idők folyamán azonban megrendült a matematikába vetett hitem. Elsősorban a kiindulópontokban és a következtetési módszerekben kezdtem kételkedni,továbbá legfőképpen abban, hogy van-e ennek valami köze a valósághoz. Végül arra az eredményre jutottam, hogy a valóságot nem lehet leírni, de ez ne is legyen a célunk; elég, ha modellünk „normális körülmények között" jó közelítését adja a várható eredménynek.

Amikor az abszolút igazságba vetett kételyemnek és célirányos matematikafelfogásomnak hangot adhattam, általában szelíd legorombításban volt részem. Egy-két éve viszont kellemes meglepetésként ért, hogy nézeteimmel nem vagyok egyedül. Először Lakatos Imre csodálatos könyvét, a *Bizonyítások és cáfolatok*at találtam meg, majd Davis és Hers könyve, *A matematika élménye* erősítette meg számomra, hogy a tudományt a kételkedés viszi előre, és semmit sem fogadhatunk el igaznak csakis azért, mert zseniális elődeink is igaznak vélték.

Ilyen indítékok alapján ismételten szemügyre vettem szűkebb szakterületemet, a számítástudományt is. Meg akarom mutatni, hogy egyes alapköveknek tekintett igazságokról különböző nézetek lehetnek, és sok minden nincs még teljesen végiggondolva.

A számítástudomány (computer science) véleményem szerint *alkalmazott műszaki tudomány*, amelybe egyaránt beletartoznak erősen elméleti (matematikai) eredmények éppúgy, mint ezek lefordítása mindennap alkalmazható receptekké, továbbá egy filozófia arról, mi e tudomány célja, melyek az eszközei, módszerei stb. Manapság, ahogy ez más tudományokban is szokásos, ez a három komponens homogén egésszé forrt össze, és egységesen lép fel a külvilággal szemben. Egyrészt ott vannak a beavatottak, akik az ismeretek birtokában igyekeznek az egyetlen igaz hitet elterjeszteni; a másik oldalon ott állnak a számítógépek felhasználói: profi, félprofi és amatőr programozók széles tábora, akik az ismereteknek csak töredékéről tudnak konkrétan, mert nincs rá idejük, nincs rá szükségük és nem is érdekli őket a „nagy egész".

Először *a szoftvertudomány egyik alapdogmáját*, az adattípus fogalmát fogjuk megvizsgálni: mit mond erről a tudomány, és hogy fest ez a fogalom a gyakorlat fényében. A μMagazin későbbi számaiban sort fogunk keríteni néhány, hasonlóan érdekes témára.

A klasszikus felfogás

Valaha minden programozó tudta, hogy a számítógépben tárolórekeszek vannak, és ezekben bitek találhatók. A programozó felelőssége volt, hogy a biteket jól kezelje. Később részben a programozók gyakorlata, részben a hardver fejlődése következtében kialakultak bizonyos absztrakt fogalmak a különböző célú memóriatartalmakkal kapcsolatban. Elvált egymástól az utasítás és az adat, az adatok különböző fajtái: egész számok, fix és lebegőpontos ábrázolású valós számok, szövegek, címek, jelzőbitek stb. Lassan *kialakultak a konvenciók*, hogyan kell ezeket használni. A szabályokat a gyakorlat szülte, és arra szolgáltak, hogy segítsék a program megírását és a hibák megtalálását. Ilyesfajta szabályokra gondolok: az utasítást menet közben nem szabad átírni, különböző adatokat ne tegyünk ugyanarra a helyre, különösen akkor ne, ha különböző típusúak stb. Ezek a szabályok ma természetesnek látszanak, akkoriban azonban a szűkös memória miatt elég nehéz volt be-

tartani őket. Időközben a gépek utasításkészlete is úgy változott, hogy ma már ezeknek a szabályoknak a megsértése vált nehézzé, ha nem éppenséggel lehetetlenné.

Az első *magas szintű programozási nyelvek* valójában ezt az állapotot rögzítették: különváltak az utasítások és az adatok, az adatokhoz rendelt típus meghatározza, hogy egy adott típusú adattal milyen műveleteket lehet végrehajtani. Például a FORTRAN-ban jól érzékelhető a nyelvi műveletek és a gépi műveletek közötti egyértelmű megfelelés.

„Élő klasszikusok"

Ma is vannak népszerű nyelvek, például maga a FORTRAN, amelyek ezt az adattípus-elképzelést követik. Ez nem meglepő, hiszen ez a filozófia egyszerű, világos, és időközben a hardver alapelvei keveset változtak, a gépek többsége ma is Neumann elvein alapszik. (Eltekintve attól, hogy Neumann Jánosnál visszatérő gondolat, hogy *a program futása közben átírja önmagát.* Ez hosszú ideig a legsúlyosabb eretnekségnek számított, és csak most kezd újra éledezni egy magasabb szinten. Erről a cikksorozatban később lesz szó.)

Az adattípus-elképzelések között sajátos zsákutcát képezett az eredeti BASIC filozófiája, amely nem tett éles különbséget az egész és a valós szám között — ahogy a matematika sem tesz —, és feltételezte, hogy az egészekkel végzett művelet eredménye is egész lesz, ha a műveletben szereplő összes szám pontosan ábrázolható. Mivel ennek megvalósítása költséges (elsősorban futásidőben), manapság a legtöbb BASIC-implementáció már — a FORTRAN-hoz hasonlóan — különbséget tesz a pontosan ábrázolt egészek és a közelítőleg ábrázolt valósok között.

Összetett szerkezetek

Az adattípusok fogalmának fejlődésében a nagy változást az ALGOL68 és a PASCAL nyelv hozta. Már a korábbi nyelvekben is voltak konstrukciók, melyekkel egyszerű elemi adatokból nagyobb összetett adatszerkezeteket lehetett létrehozni: nevezetesen tömböket és rekordokat. Az így keletkezett valamiket azonban nemigen lehetett valamilyen adattípushoz tartozónak nevezni. Két ilyen adatszerkezetnek nem sok köze volt egymáshoz. Egyrészt nem volt semmilyen közös beépített műveletük, csak az elemeikre való lebontás, másrészt *ha egy adatszerkezetre írtunk egy kezelő függvényt* vagy szubrutint, *semmi garanciánk nem volt* arra, hogy az egy *másik adatszerkezeten is helyesen fog működni.*

A két nyelv egyik nagy találmánya volt az összetett adattípus fogalma: az *elemi adattípusokból új összetett adattípus* hozható létre. Egy ilyen új típusból adatpéldányok készíthetők, amelyek egymással cseresza-batosan viselkednek: az egyiket a másiknak értékül lehet adni; a függvények pedig nem egy formális adatszerkezetet kezelnek argumentumként, hanem egy adattípust, és az argumentum helyére híváskor csak az adott típushoz tartozó objektum kerülhet. Így az összetett adatokra is igazzá vált, hogy csak az engedélyezett műveleteket lehetett végrehajtani rajtuk.

Mivel az összetett adattípus is része lehet egy még nagyobb összetett adattípusnak, így potenciálisan végtelen sok adattípus áll rendelkezésünkre.

Stílusváltás

A másik fontos és mély gondolat a programozás stílusára vonatkozott. Korábban a feladat megoldásában a döntő elem az eszköz, azaz a számítógép, illetve a programozási nyelv volt.A feladat megoldása úgy kezdődött, hogy eldöntöttük, milyen mennyiségek szerepelnek majd a programban és hogyan ábrázoljuk őket; ezután jött az algoritmus megírása. Az új felfogás szerint *a feladat logikájából kell kiindulni.* Először meg kell fontolnunk, hogy a program milyen különféle absztrakt adatféleségeken dolgozik, és ezeken milyen absztrakt tevékenységeket kell végrehajtania. Ha egy ilyen magasabb absztrakt szinten már látjuk a feladat megoldását, akkor elkezdődhet a megoldás konkretizálása. *A konkrét program megírása során eldöntjük, hogy milyen* gépi adattípussal ábrázoljuk az absztrakt adattípust, megírjuk a szükséges absztrakt műveleteket egy-egy alprogramban, és leírjuk a teljes programot az *absztrakt műveletek segítségével.*

Már az első lépés során el kell nevezni az absztrakt adattípusokat és műveleteket, majd ezekkel a fogalmakkal kell a problémát megoldani. Az így megírt programnak az az előnye, hogy a feladatmegoldás fő gondolatmenete nem keveredik össze azokkal az implementációs döntésekkel, amelyeket gyakran csak a konkrét gép ismeretében hoztunk. *A program könnyebben átvihető lesz más gépekre;* a program kisebb, önállóan tesztelhető darabokra bomlik; ha kiderül, hogy az adatok ábrázolását vagy az absztrakt műveleteket bármilyen okból meg kell változtatni (pl. ellenőrzéseket kell beiktatni az adatokra, vagy éppen ellenkezőleg; az ilyesmitől a hatékonyság érdekében meg akarunk szabadulni), akkor ez könnyen megtehető.

Ha például egy gabonaelosztási és -kiszállítási programot írunk, akkor a feladat megoldásakor elég olyan fogalmakkal dolgozni, mint a búza mennyisége és a teherautó befogadóképessége. A program felsőbb, absztraktabb szintjéhez ezeket a mennyiségeket a TÖMEG absztrakt típusba soroljuk, és az már implementációs döntés lesz, hogy ezt kilóban mérjük és lebegőpontos számmal ábrázoljuk-e vagy a zsákok számával — egészként.

Az absztrakció problémája

Az új felfogás a szakemberek körében osztatlan elismerést váltott ki. Az első lelkesedési hullám csillapultával azonban a két említett nyelvvel kapcsolatban részben implementációs, részben „filozófiai" kérdések sora merült fel. A kérdés ez volt: ha van két különböző típusú, de azonos gépi ábrázolású változóm, értékül lehet-e adni ezeket egymásnak, össze lehet-e adni őket, össze lehet-e szorozni őket; egyáltalán milyen műveletek vannak engedélyezve egy ilyen új típusú mennyiségre. Ma már tudjuk a helyes választ: *az absztrakt típusokban és az absztrakt műveletek szintjén a típuskeveredés* káros, veszélyes, *megengedhetetlen; az implementációs szinten* viszont *elkerülhetetlen:* különben nem, vagy csak igen nehezen tudjuk leírni az absztrakt műveleteket.

A két nyelv ebben a kérdésben kétféle módon foglalt állást (természetesen csak jóval később, a probléma hosszú vitája után). A PASCAL-hívők a típusok név szerinti azonosítása mellett döntöttek: két mennyiség akkor adható értékül egymásnak, ha ugyanabba a típusba tartoznak, a függvény akkor hívható meg egy aktuális paraméterrel, ha ugyanabba a típusba tartozik, mint a formális paraméter. Mivel azonban ez a felfogás jóval később kristályosodott ki, mint maga a *PASCAL nyelv,* az még nem ezt a felfogást követi, hanem *a bevezetett típusnevet* a definícióban szereplő *típusleírás szinonimájának tekinti.* A különböző implementációk kissé el is térnek egymástól a típuskompatibilitás megítélése tekintetében.

Az ALGOL68 más utat követett: *azt mondta, hogy két változó azonos típusú, ha struktúrájuk megegyezik* egymással. Itt tehát értékül lehet adni egymásnak két rekordot, ha az egyik az A nevű egész mezőből és a B nevű valós mezőből áll, és a másik fordított sorrendben, a B valós és az A egész mezőből áll, mert a struktúra megítélésében a mezőnév és a típus azonossága releváns, a mezők sorrendje pedig nem. Nem adható viszont értékül egymásnak egy kételemű egész tömb és egy két egész mezőből álló rekord. Mint látható, a típusazonosság vizsgálata ALGOL68-ban eleve igen komplikált, a pointereket tartalmazó rekurzív típusok esetén pedig még inkább.

Az idő a PASCAL-hívőket igazolta: *manapság minden valamirevaló nyelv név szerinti típusazonosságot követel meg.*

FARKAS ERNŐ

Közületek, figyelem!

Mikroszámítógépet akarnak vásárolni?

Tájékozódjanak előtte a naprakész piaci helyzetről!

Díjtalan ismertető:
MESZ
Számítástechnika
1368 Budapest
Pf. 193.

Spectrum a laborban

A Magyarregula '87 szakkiállításon jártamban, a sok bonyolult és a figyelmet megragadó mérőműszer és híradástechnikai berendezés között megakadt a szemem egy látszólag egyszerűn. Dobozba épített Spectrum billentyűzetét láttam; egyik oldalán szörcsögő, pufogó valami, a másik oldalán nyomtató, amely grafikonokat és képleteket ír.

„Automatikus mintaadagoló" — olvasom a táblán a megnevezést.

— Hogyan kerül a Spectrum egy laboratóriumi berendezésbe? — kérdeztem Pap Lászlót, a berendezés feltalálóját.

— A szigorú minőség-ellenőrzési előírások — válaszolta — igen nagy terhet rónak a laboratóriumi dolgozókra, különösen a gyógyszergyártásban. A műszerek kézi kiszolgálása, az eredmények kiértékelése lassú és fáradságos munka. Körülbelül tíz éve foglalkoztat a gondolat: hogyan lehetne a meglévő műszereket utólag automatizálni. Első lépésként egy interfészt terveztem, és kerestem a megfelelő mikroprocesszort. Az interfész két darab 8255-ös port beépítésén alapszik. Egyaránt alkalmas analóg és BCD jelek érzékelésére, átalakítására, és természetesen biztosítja a „kommunikációt" a számítógéphez kapcsolt berendezésekkel. A vezérlést és automatizálást végző mikroprocesszort először saját fejlesztéssel próbáltam elkészíteni, de nem sikerült.

Az első megfelelően működő készüléket PC-XT számítógéppel valósítottam meg. A PC azonban nagyon drága, és nem is volt szükség ilyen nagy tudású gépre: olyan felesleges volt ez, mint ágyúval verébre lőni. Akkor támadt az az ötletem, hogy megpróbálom a Spectrumot. A gép kapacitása, billentyűzete és praktikus beépíthetősége kiválóan megfelelt a célnak.

— Azt olvastam a prospektusban, hogy a beépített számítógép önállóan is használható.

— Ez így igaz. Mivel azonban a készüléket általában két műszakban, folyamatosan üzemeltetik, nincs sok idő egyéb alkalmazásokra. Nagy előny viszont, hogy a gép önállóan programozható. A felhasználónak lemezen és a biztonság kedvéért kazettán is átadjuk a programot, amely két részből áll. Az első, gépi kódban írt rész a központi egységhez kapcsolt berendezéseket vezérli. Itt megadjuk a hivatkozási címeket, de az algoritmus megváltoztatását nem javasoljuk. A rutin másik része a mérési eredmények kiértékelésére, kiíratására szolgál. BASIC nyelven készült, hogy az alkalmazó saját igénye szerint változtathassa. A programbetöltés gyorsítására a lemezmeghajtót beépítettük a központi egységbe, de „vésztartaléknak" meghagytuk a kazettaolvasó bemenetet is.

— Milyen a berendezés fogadtatása?

— Itt is tapasztalható az általános felhasználói sznobizmus. Gyakran hallom a megjegyzést: „Ez *csak* egy Spectrum".

Nos, a beszélgetés után bármily kritikusan is szemléltem meg újból az adagolót, nagyon tetszett az olcsó, praktikus megoldás. A feltaláló a lehető legegyszerűbben automatizálta a megszokott analitikai műszereket úgy, hogy az új termék adatfeldolgozásra is képes lett.

Ezen a szakkiállításon, ahol elsősorban méréstechnikai eszközöket mutattak be, másutt is jelen volt a mikroszámítógép. Nemigen maradhat a közeli jövőben sem piacon az a cég, amelyik „csupán" nagy pontosságú, fejlett mérőműszereket gyárt. Fejlesztési követelmény, mert a felhasználók természetes igénye a mikroszámítógéppel való kapcsolat.

PINKE GYÖRGY

COMMODORE 64

Hasznos programok az US

Az alábbi gépi kódú program az USR függvény felhasználásával működik.

A függvény hivásakor a zárójelek közt levő kifejezés értéke (lehet sztring is) betöltődik a lebegőpontos akkumulátorba, és ezt gépi kódú programjainkban paraméterként szerepeltethetjük. A vezérlés a $0310 címen lévő JMP utasításra kerül. Az assembly rutinunkban új értéket adhatunk a FAC-nak, s majd az RTS utasítás után a vezérlés a függvények kiértékeléséhez tér vissza, ahol még egy numerikus ellenőrzés is lesz. Ezt követően a FAC-ban lévő érték átadódik a BASIC programnak. Ha szöveges változóval térünk vissza, két PLA utasítással ki kell venni a veremből a visszatérési címet, hogy a numerikus ellenőrzést elkerüljük.

Az első programmal (*1.* és *2. lista*) a BASIC és az operációs rendszer ROM alatti címeit, valamint a karaktergenerátort tudjuk olvasni BASIC programból. A gépi kódú program a FAC-ot címformátumra konvertálja, letiltja a megszakításokat, a BASIC és KERNAL ROM-ot ki-, a CHAR ROM-ot pedig bekapcsolja. Kiolvasás és visszakapcsolás után az értéket a FAC-ba tölti. A program előnyös lassú működésű adattároláskor, vagy például SIMON'S BASIC grafikus képernyőtartalom olvasásakor $E000 fölött (lásd Softcopy 86/11-12. szám).

```
100 REM  ROM ALATTI CIMEK OLVASASA
110 REM  ERTEK = USR ( CIM )
120 REM  HIVASA: SYS 680
130 :
140 FORK=680TO711:READ A
150 POKEK,A:S=S+A:NEXT
160 IFS)<3238THENPRINT"ADATHIBA":END
170 SYS680
180 :
190 DATA169,179,160,  2,141, 17,  3,140
200 DATA 18,  3, 96, 32,247,183,160,  0
210 DATA162, 49,120,134,  1,177, 28,162
220 DATA 55,134,  1, 88,168, 76,162,179
```

1. lista

A második program a C64 és VC20 közötti szalagos adat- és programcserét teszi lehetővé (*3.* és *4. lista*).

A VC20-as számítógép óraütteme nagyobb a Commodore 64 üteméné, ezért a magnetofonszalagra eltérő frekvenciával rögzitik a jeleket. Tapasztalataim szerint a VC20 be tudja olvasni a C64-szalagokat, de fordítva nem. A közölt gépi kódú program SYS 52000 parancs után átmásolja az interpretert és az operációs rendszert a RAM-ba, majd módosítja a szalagolvasás és a szalagra írás időállandóit. A PRINT

```
USR(CIM)

1000                100         ; ROM ALATTI CIMEK OLVASASA:
1000                110         ; ERTEK = USR ( CIM )
1000                120         ; HIVASA: SYS 680
1000                130         ;
B7F7                140 FACADR  =$B7F7
B3A2                150 YTOFAC  =$B3A2
0311                160 USRVEK  = 785
0014                170 CIM     =$14
1000                180         ;
02A6                190         *=680
02A6 A9 B3          200         LDA #USR<
02AA A0 02          210         LDY #USR>
02AC 8D 11 03       220         STA USRVEK
02AF 8C 12 03       230         STY USRVEK+1
02B2 60             240         RTS
02B3 20 F7 B7       250 USR     JSR FACADR
02B6 A0 00          260         LDY #0
02B8 A2 31          270         LDX #%00110001
02BA 78             280         SEI       ;KARAKTER ROM BE
02BB 86 01          290         STX 1     ;BASIC+KERNAL KI
02BD B1 14          300         LDA (CIM),Y
02BF A2 37          310         LDX #%00110111
02C1 86 01          320         STX 1     ;VISSZAKAPCSOLAS
02C3 58             330         CLI
02C4 A8             340         TAY
02C5 4C A2 B3       350         JMP YTOFAC
02C8                360 VEGCIM .END

ZEILEN:27   SYMBOLE:6   FEHLER:0

CIM   =0014  FACADR=B7F7  USR   =02B3   USRVEK=0311  VEGCIM=02C8  YTOFAC=B3A2
```

2. lista

3. lista

```
100 REM   C64-VC20 SZALAGUZEMMODOK
110 REM   HIVASA:      SYS 52000
120 REM   C-64:        PRINT USR(1)
130 REM   VC20:        PRINT USR(-1)
140 REM   LEKERDEZES:  PRINT USR(0)
150 :
160 FORK=52000TO52200:READ A
170 POKEK,A:S=S+A:NEXT
180 IFS)<26294THENPRINT"ADATHIBA":END
190 SYS52000
200 :
210 DATA 32,134,203, 32, 82,203,169,186
220 DATA168,203, 32, 36,171,169, 56,160
230 DATA203,141, 17,  3,140, 18,  3, 96
240 DATA 32, 43,188,240,  8,144,  3,169
250 DATA 53, 44,169, 55,133,  1, 32, 82
260 DATA203,104,104,169,186,160,203, 76
270 DATA135,160,165,  1, 41,  2,208,  3
280 DATA162,  7, 44,162,  3,160,  4,189
290 DATA126,203,153,186,203,202,136,208
300 DATA246, 96, 18, 46, 46, 46, 46, 32
310 DATA 83, 90, 65, 76, 65, 71, 85, 90
320 DATA 69, 77, 77, 79, 68,  0, 67, 45
330 DATA 54, 52, 86, 67, 50, 48,169,160
340 DATA162,192, 32,176,203,169,224,162
350 DATA  0, 32,176,203,162, 11,160, 22
360 DATA185,197,203,141,168,203,185,198
370 DATA203,141,169,203,189,221,203,141
380 DATA255,255,136,136,202, 16,233, 96
390 DATA133,252,160,  0,132,251,177,251
400 DATA145,251,230,251,208,248,230,252
410 DATA228,252,208,242, 96,  1,  0,214
420 DATA253, 93,249,112,249,121,249,129
430 DATA249,155,249,176,251,174,251,210
440 DATA251,212,251,187,252, 53,229, 51
450 DATA 41, 32, 37, 16, 81,150,231,  0
460 DATA102
```

4. lista

```
100 REM  C64-VC20 SZALAGUZEMMODOK
110 REM  C64:  POKE 1,55
120 REM  VC20: POKE 1,53
130 :
140 FORK=40960TO49151:POKEK,PEEK(K):NEXT
150 FORK=57344TO65535:POKEK,PEEK(K):NEXT
160 READCIM:IFCIM=0THEN180
170 POKECIM,PEEK(CIM)+.85:GOTO160
180 POKE64466,231:POKE64468,0
190 POKE64982,229:POKE1,53
200 :
210 DATA63837,63856,63865,63873
220 DATA63899,64426,64430,64619,0
```

USR(SZAM) parancs a SZAM előjelétől függően kapcsolja ki vagy be a ROM-ot, és "C64 vagy VC20 szalagüzemmód" üzenettel tér vissza a PRINT utasításhoz. A PRINT USR(0) parancs nem kapcsol, csak az aktuális üzemmódot közli.

Mellékelve van egy BASIC-ből (70 másodpercig) másoló program is, amelynél POKE utasítással kell átkapcsolni. Itt látszik, hogy a szalagállandók 0,85-szörösükre vannak csökkentve (*5.* és *6. lista*).

A programmal hasznosíthatóak a vállalatok, intézmények VC20 számítógépei szalagos adatrögzítésre, s C64-en való feldolgozásra.

NÉMETH BÉLA

függvény alapján

```
1000              100         ;C64-VC20 SZALAGUZEMMODOK
1000              110         ;HIVASA:      SYS 52000
1000              120         ;C-64:        PRINT USR(1)
1000              130         ;VC20:        PRINT USR(-1)
1000              140         ;LEKERDEZES:  PRINT USR(0)
1000              150         ;
A81E              160 LINDUT  =$A81E
0311              170 USRVEK  = 785
BC2B              180 ELOJEL  =$BC2B
00FB              190 MUTATO  = 251
1000              200         ;
C800              210         * = 52000
C800 20 86 CB     220 SYS     JSR ROMRAM    ;BEMASOLAS
C803 20 52 CB     230         JSR BEALL
C806 A9 6A        240         LDA #SZOVEG<
C808 A0 CB        250         LDY #SZOVEG>
C80A 20 1E A8     260         JSR LINDUT    ;KIIRAS
C80D A9 08        270         LDA #USR<
C80F A0 CB        280         LDY #USR>
C811 8D 11 03     290         STA USRVEK    ;USR VEKTOR
C814 8C 12 03     300         STY USRVEK+1  ;BEALLITASA
C817 60           310         RTS
C818              320         ;
C818 20 2B BC     330 USR     JSR ELOJEL    ;USR FUGGVENY
C81B F0 09        340         BEQ NULLA
C81D 90 03        350         BCC PLUSZ
C81F A9 35        360         LDA #53       ;RAM
C821 2C           370         .BYTE$2C
C822 A9 37        380 PLUSZ   LDA #55       ;ROM
C824 85 01        390         STA 1         ;BEKAPCSOLASA
C826 20 52 CB     400 NULLA   JSR BEALL
C829 68           410         PLA           ;FUGGVENYEK
C82A 68           420         PLA           ;VISSZATERESI CIME
C82B A9 6A        430         LDA #SZOVEG<
C82D A0 CB        440         LDY #SZOVEG>
C82F 4C 87 B4     450         JMP $B487     ;FUZER A VEREMBE
C832              460         ;
C832              470         ;
C832 A5 01        480 BEALL   LDA 1         ;SZOVEG
C834 29 02        490         AND #2        ;BEALLITASA
C836 D0 03        500         BNE C84
C838 A2 07        510         LDX #7        ;*VC20*
C83A 2C           520         .BYTE$2C
C83B A2 03        530 C84     LDX #3        ;*C-64*
C83D A0 04        540         LDY #4
C83F BD 7E CB     550 BEIR    LDA TIPUS,X
C842 99 6A CB     560         STA SZOVEG,Y
C845 CA           570         DEX
C846 88           580         DEY
C847 D0 F6        590         BNE BEIR
C849 60           600         RTS
C84A 12 2E 2E     610 SZOVEG  .TEXT"▒.... SZALAGUZEMMOD"
C87D 00           620         BRK
C87E 43 2D 36     630 TIPUS   .TEXT"C-64VC20"
C886              640         ;
C886              650         ;
C886 A9 A0        660 ROMRAM  LDA #$A0      ;$A000-$BFFF
C888 A2 C8        670         LDX #$C8      ;INTERPRETER
C88A 20 D8 CB     680         JSR SUB       ;MASOLASA
C88D A9 E0        690         LDA #$E0      ;$E000-$FFFF
C88F A0 B0        700         LDX #$B0      ;OPERACIOS RENDSZER
C891 20 D8 CB     710         JSR SUB       ;MASOLASA
C894 A2 B8        720         LDX #11
C896 A0 16        730         LDY #22
```

5. lista

6. lista

```
C898 B9 C5 CB     740 READ    LDA CIM,Y     ;IDOALLANDOK
C898 8D A8 CB     750         STA POKE+1
C89E B9 C6 CB     760         LDA CIM+1,Y   ;CSOKKENTESE
C8A1 8D A9 CB     770         STA POKE+2
C8A4 BD DD CB     780         LDA ERTEK,X
C8A7 8D FF FF     790 POKE    STA $FFFF
C8AA 88           800         DEY
C8AB 88           810         DEY
C8AC CA           820         DEX
C8AD 10 E9        830         BPL READ
C8AF 60           840         RTS
C8B0              850         ;
C8B0 85 FC        860 SUB     STA MUTATO+1  ;ROM MASOLASA
C8B2 A0 00        870         LDY #0        ;A RAM-BA
C8B4 84 FB        880         STY MUTATO
C8B6 B1 FB        890 MASOL   LDA (MUTATO),Y
C8B8 91 FB        900         STA (MUTATO),Y
C8BA E6 FB        910         INC MUTATO
C8BC D0 F8        920         BNE MASOL
C8BE E6 FC        930         INC MUTATO+1
C8C0 E4 FC        940         CPX MUTATO+1  ;VEGCIM
C8C2 D0 F2        950         BNE MASOL     ;ELLENORZESE
C8C4 60           960         RTS
C8C5              970         ;
C8C5 01 00 D0     980 CIM     .WORD$00D1,$FDD6,$F550,$F376,$F379,$F981
C8D1 98 F9 AA     990         .WORD$F998,$FDAA,$FBAC,$FBD2,$FBD4,$FC5B
C8DD 35 E5 33     1000 ERTEK  .BYTE 53,$E5, 51, 41, 32, 37
C8E3 10 51 96     1010        .BYTE 16, 81,150,231,  0,102
C8E9              1020 VEGCIM .END

ZEILENI93   SYMBOLE*21   FEHLER*0

BEALL =C852  BEIR  =C85F  C64   =C85B   CIM   =C8C5  ELOJEL=BC2B  ERTEK =C8DD
LINDUT=A81E  MASOL =C8B6  MUTATO=00FB   NULLA =C846  PLUSZ =C842  POKE  =C8A7
READ  =C898  ROMRAM=C886  SUB   =C8B0   SYS   =C800  SZOVEG=C86A  TIPUS =C87E
USR   =C818  USRVEK=0311  VEGCIM=C8E9
```

ZX81

„Piszok" ellen sortörlő

A ZX81 utasításai között nem szerepel a DELETE, amellyel egyszerre több sor lenne törölhető. Ezért írtam ezt a rövid programot. Egy sort egy üres sor beírásával lehet törölni (sorszám és NEW LINE). A gép megkeresi, hogy a sor a memóriában mely címtől kezdődik, kiolvassa a sor hosszát tartalmazó két bájtot, és a címtől kezdődően ennyit és még 4 bájtot töröl. Ez a 4 bájt a sor hosszát és sorszámát tartalmazó 2×2 bájt. Ha POKE utasítással átírjuk a sorhosszt jelölő két bájtot, akkor tetszőleges számú bájtot törölhetünk. Ezen alapul a bemutatott program.

```
   1 REM ** DELETE **
   2 INPUT ETS
   3 INPUT UTS
   4 FAST
   5 LET CETS=0
   6 LET C=16509
   7 IF PEEK C*256+PEEK (C+1)>UT
S THEN STOP
   8 IF PEEK C*256+PEEK (C+1)=ET
S THEN LET CETS=C
   9 IF PEEK C*256+PEEK (C+1)=UT
S THEN GOTO 12
  10 LET C=C+(PEEK (C+2))+((PEEK
 (C+3))*256)+4
  11 GOTO 7
  12 IF CETS=0 THEN STOP
  13 LET TBS=C+(PEEK (C+2))+((PE
EK (C+3))*256)-CETS
  14 POKE CETS+2,TBS-256*INT (TB
S/256)
  15 POKE CETS+3,INT (TBS/256)
  16 LIST ETS
```

A 2. sorban az első törlendő sor sorszámát kell beadni. Ha nem létező sorszámot közöltünk, akkor a futás a 12. sorban megáll. A 3. sorban az utolsó törlendő sor sorszámát kell beírni. Ha nem létező sorszámot adunk be, akkor a futás a 7. sorban áll meg.

A program megkeresi az ETS és az UTS címét a memóriában (7—11. sorok, CETS=Címe az ETS-nak). Ezután kiszámolja a törlendő bájtok számát (13. sor), majd beírja az új sorhosszt az ETS két megfelelő bájtjára (14-15. sorok), és ETS-tól listáz (16. sor). A listában nem látható változás. Ha most a szokásos módon töröljük a kurzorral megjelölt sort, akkor az ETS és UTS soron kívül az összes közte levő is törlődik. Ha nem végezzük el a törlést és futtatjuk a programot, akkor a gép az ETS-t végrehajtja, de a következőnek az UTS utáni első sort veszi. A program önmaga törlésére is képes.

KATONA LÁSZLÓ

Relatív adattárolás

Az alábbiakban egy olyan gépi kódú programot adok közre, amely megkönnyíti a relatív fájlokkal végzett munkát.

A relatív fájlban a rekordok mezőinek kezdő pozíciója kötött (egy-egy mező mindig azonos pozíción kezdődik), és a mezők mindig ugyanakkora helyet foglalnak el. Ehhez a kiírandó adatokat azonos formátumúra kell alakítani: a PRINT # utasítással kiírva a 2 csak egy bájtot, a 65536 viszont öt bájtot foglal.

Ez a program megkíméli a programozót az átalakító rutinok megírásától. A számokat ugyanis abban a formában írja ki, ahogy azok a gép memóriájában ábrázolva vannak. Így az egész számok mindig két bájtot, a valós számok mindig öt bájtot foglalnak el. A biztonságos adatátvitel érdekében a program mindkettőhöz kiír egy ellenőrző összeget is.

A BASIC töltőprogram beírása és elindítása után, ha nem jelzett hibát, a gépi programot MONITOR üzemmódban a következő utasítással rögzíthetjük lemezre:

S "pack",8,ø6øø,ø6f3

A lemezre írás funkciót a
SYS 1536, logikaifájl-szám, változólista
utasítással érhetjük el.

Az olvasás a
SYS 1634, logikaifájl-szám, változólista
utasítással történhet.

Ha a program olvasás során hibát észlel, a FILE DATA ERROR hibaüzenetet adja. Az üzenet oka vagy a hibás pozicionálás, vagy az, hogy a lemezen levő adat típusa nem egyezik meg a számára kijelölt változó típusával (például egész változóba valós számot kellett volna olvasnia).

GNÄDIG PÉTER

```
600 rem ********************************
605 rem *
610 rem *          c-16 pack
615 rem *
620 rem ********************************
625 scnclr
630 readk:readv
640 s=1001
650 for x = k to v step 8
655 print "[clr/home]";s
660 h=0
670 for y = 0 to 7
680 read a$:a=dec(a$):h=h+a
690 poke x+y,a
700 next y
710 read a$:a=dec(a$):h=h and 255
720 if a<>h then print "hiba a";s;
    ".sorban":end
730 s=s+1:next x
740 end
1000 data  1536 , 1779
1001 data a9,00,85,d5,20,91,94,20, 68
1002 data 84,9d,86,d1,20,91,94,20, dd
1003 data a5,96,85,d2,84,d3,a5,0d, 9b
1004 data c9,ff,f0,09,a5,0e,c9,80, bd
1005 data f0,08,4c,2d,06,a2,16,4c, 7b
```

Szín

```
 1 V=49152:FOR I=V TO V+85: READ A: POKE I,A : NEXT
 2 DATA  12o, 169, 16, 160, 192,141, 20, 3
 3 DATA  140, 21, 3, 88, 96, 5, 32, 32
 4 DATA  206, 13, 192, 240, 3, 76, 49, 234,
 5 DATA  169, 5, 141, 13, 192, 234, 234, 234
 6 DATA  165, 197, 72, 72, 234, 201, 4, 208
 7 DATA  4, 238, 32, 208, 234, 104, 201, 5
 8 DATA  208, 3, 238, 33, 208, 104, 201, 6
 9 DATA  208, 22, 173, 134, 2, 72, 24, 105
10 DATA  1, 41, 15, 141, 134, 2, 104, 41
11 DATA  240, 24, 109, 134, 2, 141, 134, 2
12 DATA  76, 49, 234, 0, 0, 0, 0, 0
13 SYS/49152/: CLR : NEW
```

1. program

A program az F1, F3 és F5 funkcióbillentyűk segítségével megváltoztatja a keret, a háttér és a kiírandó karakter színét. A C64-es megszakításrutinjának kezdőcímét írja át, és emiatt az F1, F3 és F5 funkciógombok mindaddig állandóan működőképesek — ha előzőleg egyetlen alkalommal aktivizáltuk a programunkat —, amíg a számítógépet ki nem kapcsoljuk, vagy a RUN/STOP és RESTORE nyomógombokat egyidejűleg le nem nyomjuk.

Az *1. program* a BASIC lista. (Figyelem! Mielőtt futtatnánk a programot, mentsük el lemezre vagy mágnesszalagra, mert a program a RUN hatására törlődik a memóriából.) A bekeretezett szám megváltoztatásával befolyásolhatjuk annak gyakoriságát, hogy a megszakításrutin megváltoztassa a

COMMODORE 16

```
1006 data 83,86,a9,02,2c,a9,05,85, 13
1007 data d6,a6,d1,20,c9,ff,a5,d2, ac
1008 data 85,22,a5,d3,85,23,a0,00, 67
1009 data 98,48,20,b0,04,20,c0,06, 9a
1010 data 20,d2,ff,68,a8,c8,c4,d6, 63
1011 data d0,ee,a5,d5,20,d2,ff,20, 49
1012 data cc,ff,20,79,04,c9,2c,f0, 4d
1013 data ab,60,a9,00,85,d5,20,91, bf
1014 data 94,20,84,9d,86,d1,20,91, dd
1015 data 94,20,a5,96,85,d2,84,d3, 9d
1016 data a5,0d,c9,ff,f0,0a,a5,0e, 27
1017 data c9,80,f0,09,c9,00,f0,08, 03
1018 data a2,16,4c,83,86,a9,02,2c, e4
1019 data a9,05,85,d6,a6,d1,20,c6, 66
1020 data ff,a0,00,98,48,20,cf,ff, 6d
1021 data aa,68,a8,8a,91,d2,20,c0, 87
1022 data 06,c8,c4,d6,d0,ed,20,cf, 14
1023 data ff,c5,d5,d0,13,20,cc,ff, 67
1024 data 20,79,04,c9,2c,f0,af,60, 91
1025 data 18,48,65,d5,85,d5,68,60, bc
1026 data 20,cc,ff,a9,00,48,85,83, e4
1027 data 20,c9,c7,68,aa,a9,e5,85, d5
1028 data 24,a9,06,85,25,a9,00,20, 46
1029 data 5e,86,4c,d3,86,46,49,4c, 64
1030 data 45,20,44,41,54,c1,00,00, ff
1031 data 00,00,ff,ff,ff,ff,ff,ff, fa
```

tás

színeket. Ezt a változtatást elérhetjük a következő utasítások közvetlen begépelésével is: V=49152 : POKE V+25, X ahol X: 0 és 255 közé eső szám lehet. A színváltás gyakorisága akkor a legnagyobb, ha az X=1: ilyenkor másodpercenként 60-szor cserélődnek a színek. Leglassúbb a színváltás, ha X = 0, és valamelyik funkciógombot lenyomva tartjuk. Ebben az esetben kb. 4 másodpercig kell várni, hogy egy színváltás bekövetkezzen. X=5 esetében 5 * 1/60 = 1/12 másodpercenként következik be színváltás, azaz elég éppen csak egy pillanatra lenyomni valamelyik funkciógombot.

A *2. program* a SZÍNVÁLTÁS gépi kódú listája. A program a SYS(49152)-vel indítható el BASIC-ből.

SZABÓ PÉTER PÁL

2. program

```
C000   78          SEI
C001   A9 10       LDA #$10
C003   A0 C0       LDY #$C0
C005   8D 14 03    STA $0314
C008   8C 15 03    STY $0315
C00B   58          CLI
C00C   60          RTS
C00D   20 20 20    JSR $2020
```

COMMODORE 64

```
C010   CE 0D C0    DEC $C00D
C013   F0 03       BEQ $C018
C015   4C 31 EA    JMP $ EA31
C018   A9 05       LDA #$05
C01A   8D 0D C0    STA $C00D
C01D   EA          NOP
C01E   EA          NOP
C01F   EA          NOP
C020   A5 C5       LDA $C5
C022   48          PHA
C023   48          PHA
C024   EA          NOP
C025   C9 04       CMP #$04
C027   D0 04       BNE $C02D
C029   EE 20 D0    INC $D020
C02C   EA          NOP
C02D   68          PLA
C02E   C9 05       CMP #$05
C030   D0 03       BNE $C035
C032   EE 21 D0    INC $D021
C035   68          PLA
C036   C9 06       CMP #$06
C038   D0 16       BNE $C050
C03A   AD 86 02    LDA $0286
C03D   48          PHA
C03E   18          CLC
C03F   69 01       ADC #$01
C041   29 0F       AND #$0F
C043   8D 86 02    STA $0286
C046   68          PLA
C047   29 F0       AND #$F0
C049   18          CLC
C04A   6D 86 02    ADC $0286
C04D   8D 86 02    STA $0286
C050   4C 31 EA    JMP $EA31
C053   00          BRK
C054   00          BRK
```

Integrált szoftver

Szövegszerkesztő II.

Átírás más gépre

```
1 GOTO8500
10 CLEAR200:IF MEM>33000 THEN CLEAR31000
:T=300 ELSE CLEAR20000:T=200
20 Bæ=STRING¤(32,32):Sæ=STRING¤(30,'='):
CH¤='ö'
21 CLS2:PRINT@195,'          POSTAAZOO
    ':M=MEM
23 FOR X=1 TO 2000:NEXT X
30 DIM M¤(T+1)
40 GOSUB 100:PRINT' UJ ADATBAAZIS? (I/N)
':GOSUB110:IF I¤='I' THEN900
90 GOSUB 500:GOTO900
100 CLS:PRINT@42,H¤:PRINT@65,S¤:PRINT:RE
TURN
110 I¤=INKEY¤:SOUND240,1
120 I¤=INKEY¤:IF I¤='' THEN120
130 IF I¤='' THEN 140 ELSE A=ASC(I¤):IF
A>96 AND A<123 THEN A=A-32:I¤=CHR¤(A)
140 RETURN
190 FOR L=1 TO 1200:NEXT L:RETURN
200 PRINT@449,'MIT VAALASZT?':GOSUB110:R
ETURN
210 PRINT@449,'NEM JOO!':SOUND120,4:GOSU
B190:RETURN
220 PRINT@L,STRING¤(31,32):PRINT@L,'';:R
ETURN
500 GOSUB 100:PRINTTAB(7)'OLVASOM AZ ADA
TBAAZIST':OPEN'I',-1,'POSTADAT'
530 IF EOF(-1) THEN CT=0:RETURN
540 LINE INPUT#-1,A¤:LD¤=A¤
550 FOR X=1 TO T:IF EOF(-1) THEN CT=X-1:
X=T:GOTO570
560 LINE INPUT#-1,M¤(X)
570 NEXT X:CLOSE:RETURN
900 GOSUB 100:PRINTCT:'REKORDOK':PRINT'A
Z ADATOK SZAAMA';T:PRINT'A LEGUTOBBI ADA
TBEADAAS: ';LD¤:PRINT'KEEREM A DAATUMOT
(PL. 87/04/15)':INPUT' =>' ';DA¤:IF DA¤='
'THEN DA¤=LD¤:M¤(0)=LD¤ ELSE M¤(0)=DA¤
910 GOSUB100:INPUT'SOROS VAGY PAARHUZAMO
S NYOMTATOO(S/P)>';I¤
920 IF I¤='S'THEN POKE &H3FF,1
1000 GOSUB 100
1010 PRINT' <U>  UJ'
1020 PRINT' <P>  ADATBAAZIS-LISTA'
1030 PRINT' <L>  CIMKEEK KIIRATAASA'
1040 PRINT' <S>  RENDEZEES'
1050 PRINT' <E>  ADATBAAZIS-KIVONAT KEESZ
ITEES'
1055 PRINT' <T>  KIS/NAGYBETUE KAPCSOLOO
':IF UC=1 THEN PRINTTAB(6)'AATALAKITOO B
E' ELSE PRINTTAB(6)'AATALAKITOO ki'
1060 PRINT:PRINT' <Q>  MENTSD EL EES VEE
GE'
1080 GOSUB 200
1100 IF I¤='U'THEN2000ELSE IF I¤='P'THEN
 3000 ELSE IF I¤='L'THEN 4000 ELSE IF I¤
='S'THEN 5000 ELSE IF I¤='E'THEN 7000 EL
SE IF I¤='Q' THEN 8000 ELSE IF I¤='T' TH
EN 1110 ELSE GOSUB 210:GOTO1080
1110 UC=1-UC:GOTO1000
1200 '
1210 GOSUB100
1220 PRINT@97,'1:';TT¤:PRINT@129,'2:';GN
¤:PRINT@161,'3:';LN¤:PRINT@193,'4:';A1¤:
PRINT@225,'5:';A2¤:PRINT@257,'6:';CS¤:PR
INT@289,'7:';PH¤:PRINT@321,'8 (KOODMEZOE
)=';CD¤:RETURN
1300 '
1320 L1¤=TT¤+' '+GN¤+' '+LN¤:L6¤=TT¤+' '
+LN¤:L2¤=A1¤:L3¤=A2¤:L4¤=CS¤:L5¤=PH¤:RET
URN
1500 R¤=LN¤+CH¤+GN¤+CH¤+TT¤+CH¤+A1¤+CH¤+
A2¤+CH¤+CS¤+CH¤+PH¤+CH¤+CD¤:RETURN
1600 GOSUB1710:L¤=LEFT¤(R¤,P-1):GOSUB17
00:GN¤=LEFT¤(R¤,P-1):GOSUB1700:TT¤=LEFT¤
(R¤,P-1):GOSUB1700:A1¤=LEFT¤(R¤,P-1):GOS
UB1700:A2¤=LEFT¤(R¤,P-1):GOSUB1700:CS¤=L
EFT¤(R¤,P-1):GOSUB1700:PH¤=LEFT¤(R¤,P-1)
:GOSUB1700:CD¤=R¤:R¤='':RETURN
1610 IF UC<>1 THEN RETURN ELSE F1=1:FOR
W=1 TO LEN(R¤)-10
1620 CH=ASC(MID¤(R¤,W,1)):IF CH<65 OR CH
>92 THEN F1=1:GOTO1650
1630 IF F1=0 AND CH<91 THEN CH=CH+32:MID
¤(R¤,W,1)=CHR¤(CH)
1640 F1=0
1650 NEXT W:RETURN
1700 R¤=RIGHT¤(R¤,LEN(R¤)-P)
1710 P=INSTR(1,R¤,CH¤):RETURN
2000 GOSUB 100:PRINT' <A>DDJ AZ ADATBAAZ
ISHOZ':PRINT' <D> TDEROELJ EGY ADATOT':P
RINT' <C>EREELJ ADATOT':PRINT' <R> VISSZ
A A FOEMENUEHOEZ'
2010 GOSUB200:IF I¤='R' OR I¤='Q' THEN 1
000 ELSE IF I¤='A'THEN 2500ELSE IF I¤='D
'THEN 2300 ELSE IF I¤='C'THEN 2100 ELSE
GOSUB 210:GOTO 2010
2100 GOSUB100:PRINT' CSERE':PRINT:PRINT'
L - AZ ADAT NEVE VAGY...':PRINT' C - A
KERESENDOE KOOD':GOSUB 110:IF I¤='L'THEN
 2105 ELSE IF I¤='C'THEN 2200 ELSE2100
2105 GOSUB100:PRINT' AZ EDDIGI NEEV'::IN
PUT N¤
2110 FOR X=1 TO CT:P=1
2120 P=INSTR(P,LEFT¤(M¤(X),10),N¤)
2130 IF P=0THEN 2180
2140 IF X>CT THEN GOSUB100:PRINT' NEM TA
LAALOM':GOSUB190:GOTO2000
2150 R¤=M¤(X):GOSUB1600:GOSUB1200:PRINT@
449,'EZ AZ(I/N VAGY Q)':GOSUB 110
2160 IF I¤<>'I' THEN 2170 ELSE GOSUB2600
:GOSUB1500:M¤(X)=R¤
2170 IF I¤='Q'THEN X=CT
2180 NEXT X:GOSUB100:PRINT' A KERESEESNE
K VEEGE':GOSUB190:GOTO2000
2200 GOSUB100:GOSUB1600:FOR X=1 TO CT
2210 R¤=M¤(X):IF R¤<>''THEN 2260 ELSE GOS
UB8200:IF HIT=0 THEN2260
2220 GOSUB1600:GOSUB1200:PRINT@449,'EZ A
Z(I/N VAGY Q)':GOSUB110
2230 IF I¤<>'I' THEN 2250 ELSE GOSUB2600
:GOSUB1500:M¤(X)=R¤
2250 IF I¤='Q'THEN X=CT
2260 NEXT X:GOSUB100:PRINT'A KERESEESNEK
VEEGE':GOSUB190:GOTO2000
2300 GOSUB100:PRINT'ADATTOERLEES':PRINT:
PRINT'A LEGUTOBBI NEEV':INPUT N¤:GOSUB10
0
2310 FOR X=1 TO CT:P=1:P=INSTR(P,LEFT¤(M
¤(X),20),N¤)
2320 IF P=0 THEN NEXTX
2340 IF X>CT THEN GOSUB100:PRINT' NEM TA
LAALOM':GOSUB190:GOTO2000
2350 R¤=M¤(X):GOSUB1600:GOSUB1200:PRINT@
449,'EZ AZ(I/N)':GOSUB110
2360 IF I¤='I' THEN L=449:GOSUB220:PRINT
'KITOEROELVE':FOR Y=X TO CT-1:M¤(Y)=M¤(Y
+1):NEXT Y:M¤(CT)='':X=CT:CT=CT-1:NEXTX:
GOTO2000
2370 NEXT X:GOSUB100:PRINT' NEM TALAALOM
':GOSUB190:GOTO2000
2500 IF CT=>T THEN1000ELSE CT=CT+1:R¤=ST
RING¤(7,'ö'):GOSUB1600:GOSUB1200:GOSUB25
20:GOSUB 2510:GOSUB2515:GOSUB2530:GOSUB2
540:GOSUB2550:GOSUB2560:GOSUB2570:GOTO25
90
2510 PRINT@417,'KERESZTNEEV':LINE INPUT'
>';LN¤:GOSUB1200:RETURN
2515 PRINT@417,'CIMZEES (PL IGAZGATO)':L
INE INPUT'>';TT¤:GOSUB1200:RETURN
2520 PRINT@417,'CSALAADNEEV':LINE INPUT'
>';GN¤:GOSUB1200:RETURN
2530 PRINT@417,'A CIM ELSOE SORA (PL A V
AALLALATNEEV)':LINE INPUT'>';A1¤:GOSUB12
00:RETURN
2540 PRINT@417,'A CIM MAASODIK SORA (PL
AZ UTCANEEV)':LINE INPUT'>';A2¤:GOSUB120
0:RETURN
2550 PRINT@417,'HELYSEEGNEEV':LINE INPUT
'>';CS¤:GOSUB1200:RETURN
2560 PRINT@417,'TELEFONSZAAM':LINE INPUT
'>';PH¤:GOSUB1200:RETURN
2570 PRINT@417,'KOODMEZOE (LEGFELJEBB 10
 HELY)':LINE INPUT'>';CD¤:CD¤=LEFT¤(CD¤+
B¤,10):GOSUB1200:RETURN
2590 GOSUB 2600:GOSUB1500:M¤(CT)=R¤:GOTO
2000
2600 PRINT@449,'VAALTOZTAT?(I, N VAGY *)
':GOSUB110
2610 IF LEFT¤(I¤,1)='N' THEN CLS:RETURN
2620 IF LEFT¤(I¤,1)='I' THEN GOSUB 1200:
PRINT@449,'MELYIK SORT?':GOSUB110
2630 I=VAL(I¤):IF I<1 OR I>8 THEN GOSUB
1200:GOSUB210:GOTO2600
2640 GOSUB1200:ON I GOSUB 2515,2520,2510
,2530,2540,2550,2560,2570
2650 GOTO2600
3000 GOSUB100:PRINT' ADATBAAZISLISTA':GO
SUB7100:OPEN'O',-2,'PRINT'
3010 GOSUB100:GOSUB8100:GOSUB100:PRINTCT
;'REKORDOKAT TALAALTAM':GOSUB 3500
3100 FOR L=1 TO CT:IF M¤(L)<>'' THEN R¤=
M¤(L) ELSE GOTO 3120
3102 IF LN>54 THEN GOSUB3130:GOSUB3500
3104 GOSUB 8200:IF HIT=0 THEN3120 ELSE P
RINT@449,'REKORD #';L
3110 GOSUB1610:GOSUB1600:GOSUB1300:TT¤=L
1¤:GOSUB8300:L1¤=TT¤:TT¤=L2¤:GOSUB8300:L
2¤=TT¤:TT¤=L3¤:GOSUB8300:L3¤=TT¤:TT¤=L4¤
:GOSUB8300:L4¤=TT¤:TT¤=L5¤:GOSUB8300:L5¤
=TT¤:TT¤=CD¤:GOSUB8300:CD¤=TT¤:PRINT#-2,
TAB(10)L1¤:PRINT#-2,TAB(10)L2¤:PRINT#-2,
TAB(10)L3¤
3115 PRINT#-2,TAB(10)L4¤;TAB(40)L5¤;'
';:PRINT#-2,CD¤:PRINT#-2,' ':LN=LN+5
3120 NEXTL:PRINT@449,'A LISTA VEEGE':GOS
UB3130:GOTO1000
3130 FOR X=LN TO 65:PRINT#-2,' ':NEXT X:
RETURN
3500 TT¤=DA¤:GOSUB8300:DA¤=TT¤:PRINT#-2,
' ':PRINT#-2,' ':PRINT#-2,TAB(10)'LISTA'
 TAB(70-LEN(DA¤))DA¤:LN=4:PRINT#-2,' ':R
ETURN
4000 TZ=0:GOSUB100:PRINT' CIMKEEK':GOSUB
7100
4010 GOSUB100:GOSUB8100:GOSUB4800
4020 GOSUB100:PRINT'A FEJLEEC SORAINAK S
ZAAMA(1-9) VAGY <P> KIIRATAAS'
4030 GOSUB110:IF VAL(I¤)>0 THENGOSUB4900
:GOTO4020
4040 IF I¤='P' THEN4100ELSE GOSUB210:GOT
O4010
4100 OPEN'O',-2,'CIMKEEK':GOSUB100:PRINT
'CIMKEKIIRATAAS'
4110 FOR LP=1 TO CT:IF M¤(LP)='' THEN 42
00 ELSE R¤=M¤(LP):GOSUB8200:IF HIT=0 THE
N 4200
4120 TT¤=CD¤:GOSUB8300:CD¤=TT¤:GOSUB1610
:GOSUB1600:GOSUB1300:IF CP¤='I'THEN PRIN
T#-2,TAB(15) CD¤ ELSE PRINT#-2,' '
4130 LX¤=L1¤:TT¤=L1¤:GOSUB8300:L1¤=TT¤:T
T¤=L2¤:GOSUB8300:TT¤=L2¤:TT¤=L3¤:GOSUB83
00:L3¤=TT¤:TT¤=L4¤:GOSUB8300:L4¤=TT¤:PRI
NT#-2,L1¤:PRINT#-2,L2¤:PRINT#-2,L3¤:PRIN
T#-2,L4¤:PRINT#-2
4140 PRINT@161,'A KIVAALASZTOTT ';LP;' R
EKORD ':PRINTLX¤
4200 NEXT LP:GOSUB190:GOTO1000
4800 GOSUB100:PRINT'A KOODSOROKAT VAGY A
 CIMKEEKET NYOMTATTATJA (I/N)':GOSUB 110
:IF I¤='I' THEN CP¤='I' ELSE IF I¤='N' T
HEN CP¤='' ELSE 4800
4810 RETURN
4900 CLS0:OPEN'O',-2,'CIMKEEK':FOR L=1 T
O VAL(I¤):PRINT#-2,TAB(15)'KOODSOR':PRIN
T#-2,'IDE IRANDOO A NEEV':PRINT#-2,'A CI
M ELSOE SORA':PRINT#-2,'A CIM MAASODIK S
ORA':PRINT#-2,'HELYSEEGNEEV':PRINT#-2:NE
XT L:CLOSE-2:RETURN
5000 GOSUB100:PRINT'RENDEZEES':N=1
5010 N=N+1:C=0:FOR X=CT TO N STEP-1
5020 IF M¤(X-1)<=M¤(X) THEN 5030 ELSE M¤
=M¤(X):M¤(X)=M¤(X-1):M¤(X-1)=M¤:C=C+1
5030 NEXTX:PRINT@93,'CIKLUSSZAAM':N:PRIN
T C;'KICSEREELEES':IF C>0 THEN 5010 ELSE
 1000
7000 CS=PEEK(329):POKE329,255:GOSUB100:P
RINT'KIVONATKEESZITEES':GOSUB8100:GOSUB1
00:PRINT'ADATBAAZISNEEV (HA NEM VAALTOZT
AT: KIVONAT)':LINEINPUT'=>';DS¤:POKE329,
CS:IF DS¤='' THEN DS¤='KIVONAT':PRINT'AZ
 ADATBAAZISNEEV';DS¤:GOSUB7100
7010 OU=0:GOSUB100:PRINT' A MAGNETOFONT
BEAALLITANI':GOSUB7110:OPEN'O',-1,DS¤:GO
SUB100
7020 FOR L=1 TO CT:IF M¤(L)<>'' THEN R¤=
M¤(L):GOSUB8200 ELSE GOTO7030
7025 IF HIT=1 THEN GOSUB1610:PRINT@-1,R¤
:OU=OU+1:PRINT@129,'A KIVAALASZTOTT ';L;
'REKORD';OU
7030 NEXT L:CLOSE:GOSUB100:PRINT'A KIVON
AT KEESZ'
7040 PRINT'LESZ MEEG MAASOLAT.....':GOSU
B7100:GOTO7010
7100 PRINT:PRINT'HA NEM Q-T AD BE VISSZA
TEER A FOEMENUEHOEZ':GOSUB 110:IF I¤='Q'
 THEN 1000 ELSE RETURN
7110 FOR X=1 TO 20:SOUND200,1:PRINT@320,
' ':SOUND200,1:PRINT@325,'A MAGNETOFON B
EKAPCSOLVA':NEXT X:MOTORON:FOR X=1 TO 44
00:NEXTX:RETURN
8000 GOSUB100:PRINT'KEESZ AZ ADATBAAZIS'
:PRINTDA¤:GOSUB 7100:GOSUB7110:OPEN'O',-
1,'POSTADAT':PRINTTAB(5)'AZ ADATBA
AZIST KIRATOM'
8010 FOR X= 0 TO CT:IF M¤(X)<>'' THEN PR
INT#-1,M¤(X)
```

Mindenekelőtt ismét utalok arra, hogy a sorozat korábbi részeiben tárgyaltakat nem ismétlem meg, egy dolog említésének kivételével: az olvasók képernyőjén a részletek elhelyezése valószínűleg eltér az általam használt DRAGON-étól, melynél a képernyő 16 soros, soronként 32 karakterrel. (Megjegyzem, hogy a sorozat első részében tévesen szerepel az EOF kulcsszónál a kettős kereszt jel.)

A program a 20-as sorában azt vizsgálja, hogy 48 vagy 32 k szabad memóriaterületű DRAGON típussal dolgozunk-e. Ettől függ a T változó értéke: a programbetöltés után szabadon maradt bájtérték századrésze legyen.

Az 500-as sorban a 329. tárolóhelyre írt Ø szerepe a párhuzamos illesztésű nyomtató kiválasztása. Más gépeknél ez nem szükséges.

A 3080-as sorbeli AND operátor sok BASIC-változatban hiányzik. Ezeknél a következő programrészlettel helyettesíthető:

```
3080 NN=VAL(A$):IF NN<=Ø THEN
     3090
3085 IF NN>T THEN3090
3087 N=NN:GOTO3100
```

Ugyanígy helyettesíthető a 3320, 3490, 4070, 7520, 7780, 7785, 8110-es sor.

A 3300-as sorban a POKE sorozat a billentyűzetmátrix-állapottáblát írja tele. A 255 értékek azt jelentik, hogy nem nyomtuk le a billentyűket.

A 4240, 4250, 5000, 6110, 7000, 7010, 7020, 7100, 7120, 7140, 7410, 7470, 8000-es sorban levő IF... THEN... ELSE... utasítássor sok BASIC-változatban hiányzik. A helyettesítő sok megoldásban az ELSE előttiek végére egy olyan ugróutasítást kell írni, amely az „igaz" ág utáni folytatást biztosítja. (Ennél a sornál a következő sor.) Az ELSE szó természetesen elmarad, és az utána következő rész új sorba kerül, melynek végén a „nem — igaz" ág utáni folytatásra ugró utasítás következik, ha nem a következő sorban lesz a folytatás.

A 8110-es sorban levő OR operátor helyettesítésére szolgáló részlet (a sorban az AND előttieket figyelmen kívül hagyva):

```
8110 IF LEN(A$)<LN-7 THEN RE-
     TURN
8112 IF LEN(TS)<LN-12 THEN RE-
     TURN
```

DR. SIMONYI ENDRE

```
9020 NEXTX:CLOSE:RESTORE:GOSUB100:PRINT:
PRINT"HA NEM Q-T AD BE UJABB MAASOLATOT
KEESZIT":GOSUB110:IF I$<>"Q" THEN 8000
9030 PRINT:END
8100 PRINT@129,"AZ EGEESZET (I/N)";:GOSU
B110
8110 IF I$="I" THEN SL$="MIND":RETURN EL
SE SL$=""
8130 PRINT:PRINT"BEADANDOO VAGY A KOOD V
AGY <ENTER> AMIRE BEADHATOO AKAARMI":LIN
E INPUT" )";M1$
8140 PRINT"RENDBEN (I/N)";:GOSUB110:IF I
$="I"THEN RETURN ELSE GOSUB 100:GOTO 810
0
8200 HIT=0:IF SL$="MIND" THEN HIT=1:RETU
RN
8210 CD$=RIGHT$(R$,10):FOR SL=1 TO 10-LE
N(M1$)
8220 IF MID$(CD$,SL,LEN(M1$))=M1$ THEN H
IT=1:SL=10
8250 NEXT SL:RETURN
8300 ZY$="":FORXX=1TO LEN(TT$):ZX$=MID$(
TT$,XX,1):IF (ZX$<>"%")AND(ZX$<>"#")AND(
ZX$<>"&")AND(ZX$<>"@") THEN8350
8310 IF ZX$="@" THEN ZX$=CHR$(125):GOTO8
350
8320 IF ZX$="%" THEN ZX$=CHR$(124):GOTO8
350
8330 IF ZX$="#" THEN ZX$=CHR$(96):GOTO83
50
8340 ZX$=CHR$(126)
8350 ZY$=ZY$+ZX$:NEXTXX:TT$=ZY$:RETURN
```

A VC—1541 lemezegység furcsa titkai

Ez-az a katalógusokkal

Mint közismert, az 1541-es meghajtó a lemez 18-as sávját használja katalógus céljára. A Katalógus a LOAD"S", 8 parancscsal a memóriába tölthető, LIST-tel pedig kilistázható.

A következő csalafintaságokkal a listázásban érdekes — néhol nem is haszontalan — hatásokat érhetünk el. A lemezegységgel való elmélyült foglalatossághoz elengedhetetlenül szükséges egy jó minőségű lemezmonitor is. (Az általam ismertek közül a legalkalmasabb az EXDOS—DISK-DOCTOR.)

A katalógus listázásának megakadályozása

A listázó rutin számára köztudottan a nulla bájt jelenti a listázás végét. Ez érvényes a katalógus listázására is.

A lemez neve (max. 16 karakter) a 18-as sáv 0-dik szektorának 144-dik bájtján kezdődik. Ha tehát innen kezdve beírjuk a három nullát, a feladatot máris megoldottuk. Nem valami elegánsan ugyan, mert a képernyőn a listázás után megjelenik egy 0 és invertáltan „R".

A frappáns megoldás: először három DEL karaktert kell beírnunk, és csak utána jöhet a három nulla bájt. (A DEL karakter ASCII kódja 20.)

Sajnos még ez sem biztosít abszolút védelmet a listázás ellen, mert pl. a HELP PLUS "S" parancsára nincs hatással.

Adott fájl utáni listázás megakadályozása

A dolog itt is a három nulla bájton alapul. Álljunk rá az adott fájl katalógusbejegyzésére, pontosabban a fájl nevére! Ha a név hossza kisebb 13 karakternél, akkor a név utáni első 160-as kódú karaktertől kezdve írjuk be a három nullát. Ha hosszabb, akkor bizony a nevet meg kell csonkítanunk.

Fájlnév kiegészítése ",8," vagy ",8,1" toldattal

Gyakorlottabb programozók és felhasználók kedvelt szokása, hogy a képernyőn már meglévő feliratokat nem írják be újra, hanem a kurzorral rámennek a kívánt sorra, esetlegesen javítják, végül „elküldik". Ennek talán legismertebb példája a katalógusról való töltés, mikor is a betölteni kívánt fájl neve elé "10" (a LOAD rövidítése), neve után pedig a toldat kerül.

Ismeretlen programok betöltéséhez célszerű mindig a ",8,1" alakot használni. Akinek már gazdag programgyűjteménye van, nagy valószínűséggel nem tartja fejben az összes másodlagos címet. Egyszerűbb és elegánsabb dolog, ha a katalógust listázva, a megfelelő toldatok már benne vannak a listában. Ennek kivitelezése azon alapul, hogy egy bejegyzésnél — ha a fájlnév rövidebb, mint 16 karakter — a DOS shiftelt szóközökkel (kódja 160) egészíti ki 16-ra. A listázás során az *első* 160-as kódú karakternél írja ki a listázó rutin a második (a fájlnevet lezáró) idézőjelet — a többi shiftelt szóközt pedig nem veszi figyelembe.

A megvalósítás során a fájl nevét követő *második* 160-as karaktertől kezdve írjuk be a toldat karakterkódjait, nem elfeledkezve a max. 16 karakterhosszról.

Az ily módon kezelt katalóguslistánkra már csak a LOAD-ot kell beírnunk (valamint a RETURN-t megnyomnunk).

Természetesen ha van hely, az ominózus három nulla bájtot is beírhatjuk. Ez akkor célszerű, ha több összetartozó (program-) fájlunk van a lemezen és csakis az indító program nevét akarjuk a listán megjeleníteni. Ez esetben viszont az indító programkatalógus bejegyzése fizikailag meg kell, hogy előzze a többiekét.

A lemez ID-jének módosítása

Talán nem mindenki előtt ismert, hogy a lemez azonosítására öt karakter használható, melyek a listázásnál megjelennek. A blokkokba ugyan a formattáláskor megadott kétkarakteres ID lesz felírva, de a 18-as sáv 0-dik blokkjának 162—166-os bájtjai tetszőlegesen átalakíthatók.

Ezen öt karakter helyére bármilyen 0—255 kódú karaktert írhatunk be. 32 alatti kóddal (pl. DEL=20, HOME=19, RETURN=13) megkeverhetjük a listázó rutint. A 128 és az a fölötti kódokat a listázó rutin tokenként értelmezi és a megfelelő BASIC alapszó lesz kiírva. 32—127 között a „hagyományos" karakterek íródnak ki.

Lemezre írás megakadályozása leragasztás nélkül

Nem szeretnénk, hogy preparált katalógusunkba véletlenül beletöröljünk.

Ha a 18-as sáv 0 szektor 2-ik bájtját 65-ről 66-ra írjuk át, akkor hagyományos módon a lemezre már nem írhatunk, és csak az újraformattálás segít.

KÁNTOR GYÖRGY

ADOM A MAGYARÁZATOT!

Nagy Szabolcs olvasónk a következőket írja:

„Spectrum gépem van, és programozgatása közben rájöttem egy hibájára (legalábbis feltételezésem szerint az). A gép ugyanis rosszul hatványozza a negatív számokat, ha a hatvány páros szám. Ha így írom be:
PRINT −2↑2, az eredmény −4
ha pedig így:
PRINT (−2)↑2,
akkor INVALID ARGUMENT a gép válasza."

A jelenség oka nem géphiba. Minden más Spectrum és a legtöbb mikroszámítógép BASIC fordítóprogramja így működik.

Mi is történik itt? Olvasónk rájött arra, hogy két teljesen különböző dolog (sőt azt hiszi, hogy három, hiszen a nem páros hatványokra nem állítja a hibás eredményt), attól függően, hogy zárójelez vagy sem.

Tárgyaljuk külön a két esetet, mert kétféle okot kell megtalálnunk.

Az első eset magyarázata a műveletek prioritása. A BASIC fordító általában egy kifejezésen belül balról jobbra haladva hajtja végre a műveleteket, ha azok egyenlő prioritásúak. Mielőtt ezt tenné, kikeresi — szintén ebben az irányban haladva — a legmagasabb prioritású, azt végrehajtja, majd ezt ismétli. A számítási műveletek között a legmagasabb prioritási szintű a hatványozás, a következő szint a szorzás és osztás, azután jön az összeadás és kivonás. A zárójelezés ettől való eltérést eredményezhet; most a zárójelbe tett rész egy kifejezés, így a fordító azon belül külön vizsgál. Az első esetben a fordító először hatványoz, és utána az eredményt kivonja zérusból (ezt jelenti a negatív előjel). Ezt, mint a későbbiekben látjuk, az érvényes számtartományon belül a fordítóprogram mindig végre tudja hajtani.

A második esetben a zárójellel arra kényszerítjük a fordítót, hogy negatív számot hatványozzon (ezt is akartuk!). Ez azonban közvetlenül nem megy. Miért? A hatványozást a fordítóprogramok a következő átalakítással hajtják végre:
A↑B = EXP(LOG(A↑B)) =
EXP(B*LOG(A)). Negatív szám logaritmusát viszont nem képezheti, így hibaüzenettel leáll.

Munkánk során gyakran előfordul, hogy olyan számot kell hatványoznunk, amelynek előjelét előre nem ismerjük. Ilyenkor használható az alábbi programrészlet:

```
10 IF A<0THEN
A=−A:Y=A↑B:Y=−Y:GOTO30
20 Y=A↑B
30 ...
```

* * *

Az 1987/6-os számunk 41. oldalán Tóth József a következőt kérdezte: mi az oka annak, hogy ha Atari 800XL típusú gépén SAVE paranccsal BASIC programot tárol, majd LOAD (vagy CLOAD) paranccsal tölti vissza, a visszavitt anyag hosszabb, mint a kiküldött?

Ez az ATARI—BASIC egyik hibája. Az ATARI—BASIC minden betöltésnél 16 bájtot hozzáír a program végéhez. Ez elkerülhető, ha LIST és ENTER-be dolgozunk.

Dr. S. E.

Legyünk igényesek!

A Magazinban megjelent felhívás szerint olyan vállalkozókat keresnek, akik a teljes magyar ékezetes karakterkészlet nyomtatását meg tudják oldani.

Ezúton jelezzük, hogy a Seikosha SP180 VC jelű, jelenleg 24 300 forintért kapható, levélminőségben is írni képes nyomtatót, valamint az igen elterjedt MPS 801-et cégünk (Video Elektronika GMK, 1475 Bp., Pf.: 142. Tel.: 113-914) át tudja alakítani úgy, hogy ennek a követelménynek megfeleljen.

Átalakítás után az SP180 VC egyéb szolgáltatásai (vastag, dőlt, nyújtott, aláhúzott stb. betűk) bonyodalmas szoftver módszerek nélkül, az eredeti nyomtatási sebességgel, levélminőségben állnak rendelkezésre. (Levélminőségű ékezetes írást szoftver módszerrel nem lehet elérni!)

Az átalakított nyomtató például a C64 vagy a C Plus/4 megfelelő ékezetes szövegszerkesztőjével minden igényt kielégítő üzleti levelezésre, szerkesztésre alkalmas.

Az MPS 801 nyomtatóval az írás képe szerényebb lesz (egyes kisbetűk szára nem lóg le, nincs levélminőségű írás, kevesebb az egyéb szolgáltatás), de a magyar helyesírásnak megfelelő szöveg azon is előállítható.

Megrendelés esetén más (cirill, görög, speciális) karakterek beépítését is vállaljuk.

Tájékoztatásul közöljük, hogy az átalakítás egy nap alatt elkészül; ára az említett printereknél 1700,— Ft, külön igények esetén némi felárral.

DR. TOLNAI JÁNOS
közös képviselő

A tartalomleírások az alábbi folyóiratokban megjelent programlistákról készültek:

A folyóirat neve	Kódja
64'er Magazin	64er
Chip Magazin	chip
Commodore Horizons	coho
Commodore Microcomputers	comi
Compute!	cute
Computer Persönlich	pers
Happy Computer	happ
hc - Mein Home-Computer	hc
mc - Zeitschrift	mc
Run /USA/	run
Sinclair User	sinc
Your Sinclair	ysin

A tartalomleíró szóláncokat permutáltuk, a szólánc variánsokat pedig alfabetikusan rendeztük.

A tartalomleírás egy szóláncból áll, majd a listában ezt követi a forrás megjelölése a folyóirat azonosítójával, a megjelenés dátumával és a cikk előkereséséhez a kezdő oldalszám és a terjedelem megadásával. A mellékelt lista értelmezéséhez még az alábbiakat kell tudni. A tartalomleírás szóláncában elsőként a téma átfogó megnevezése, utána a számítógéptípus(ok), ezt követően a szűkebben jelölt tartalom meghatározása szerepel, majd esetlegesen néhány, a közleményt minősítő adat (például : cikksorozat).

A forráshely karaktersorozatát nyíl vezeti be, melyet a / jelig a folyóiratok azonosítója, a két / jel között az évszám, folyóiratszám és kötőjellel a kezdő oldalszám követi, a végén pedig a közlemény teljes oldalterjedelme áll.

A folyóiratok a SZÁMALK szakkönyvtárában (Budapest XI., Szakasits Á. út 68. Nyitva: 8-tól fél 5-ig. Tel.: 853-111/251) is föllelhetők. A kiválasztott anyagról másolat rendelhető az alábbi formában:

SZÁMALK Szakkönyvtára
Budapest, 112. Pf.: 146. 1502

Megrendelem a Mikroszámítógép Magazin 1987/ sz. alapján a következő folyóirat-oldal-másolatokat:

Kód: _____ Példányszám: _____
Kód: _____ Példányszám: _____
Kód: _____ Példányszám: _____

A megrendeléshez csatolom az oldalankénti 8,- Ft-os szolgáltatási díj befizetését igazoló csekkszelvényt.
Dátum, név, pontos cím.

PROGRAMLISTA
adatkezelés▯commodore 64▯Keszletgazd alkodas▯indexszekvencialis technika ->run2/86.04-52/8

PROGRAMLISTA
adatrendezes▯commodore 128▯(quicksort algorithmus) ->hc/86.06-48/3

PROGRAMLISTA
adatrendezes▯commodore 64▯rendezes csak Kiirataskor ->run2/86.04-84/5

PROGRAMLISTA
apple ii▯atari 400/800 xl/xe▯commodore 64▯128▯jatekprogram▯(miami ice) ->cute/86.06-34/6

PROGRAMLISTA
apple ii▯commodore 64▯compute!▯(mlx) ▯listabegepelesi segedlet ->cute/86.06-120/4

PROGRAMLISTA
applesoft▯matematika▯Komplex szamok beepitese a basic interpreterbe ->mc/86.06-84/4

PROGRAMLISTA
atari 400/800 xl/xe▯Keptarolas Koala pad/micropainter formatumban ->happ/86.06-84/5

PROGRAMLISTA
atari 400/800 xl/xe▯vertikalis mozgas▯autostart▯logikai Kapcsolas ->hc/86.06-51/2

PROGRAMLISTA
atari 520 st▯barKacsolas▯commodore 64▯128▯nyomtato(centronics)▯KabelKeszites▯vezerlo szoftver ->happ/86.06-154/3

PROGRAMLISTA
atari▯compute!▯osszegellenorzo rutin listabegepeleshez ->hc/86.06-69/1

PROGRAMLISTA
barKacsolas▯helyesbites▯(86.03)57/(86.04)38/48/(86.05)105/147 ->64er/86.06-73/1

PROGRAMLISTA
basic programozas▯commodore 64▯szubrutinKonyvtar felhasznalasa▯atszamozas ->run2/86.04-89/3

PROGRAMLISTA
botKormany▯commodore 64▯eger▯(mouse)▯eger szimulalas ->run2/86.03-46/3

PROGRAMLISTA
cikKsorozat▯commodore 128▯fuzerKezelés▯(stringcopy) es (stringswap) rutin ->happ/86.06-54/4

PROGRAMLISTA
cikKsorozat▯commodore 64▯jatekprogram Keszites▯grafikus editor ->happ/86.06-51/3

PROGRAMLISTA
cikKsorozat▯commodore 64▯programok osszeKapcsolasa ->cute/86.06-74/3

PROGRAMLISTA
commodore 128▯80 Karakteres soros nyomtatasa ->run2/86.03-83/6

PROGRAMLISTA
commodore 128▯KarakterKeszlet modositas▯a 8563-processzor mukodesmodja ->run/86.06-52/4

PROGRAMLISTA
commodore 64▯4 Kbyte ram-tobblet interpreter athelyezessel ->64er/86.06-79/2

PROGRAMLISTA
commodore 64▯ablakok es gorgetheto menu generalasa ->cute/86.06-78/5

PROGRAMLISTA
commodore 64▯basic bovites Kepernyokezeleshez ->run2/86.03-79/4

PROGRAMLISTA
commodore 64▯basic rutinok szekvencialis fileKent valo tarolasa ->run/86.06-84/3

PROGRAMLISTA
commodore 64▯fileblokKok atlapozasa▯Kepernyo es ascii Kod megjelenitese ->run2/86.04-50/2

PROGRAMLISTA
commodore 64▯funKciobillentyu-tobblet szoftver utjan ->run2/86.04-42/2

PROGRAMLISTA
commodore 64▯grafika▯100 szabadon valaszthato szines sor ->run2/86.03-63/12

PROGRAMLISTA
commodore 64▯grafika▯interaktiv szovegbeiras ->run2/86.03-98/6

PROGRAMLISTA
commodore 64▯grafikai resz Kikeresese az operativ tarbol▯nyomtato rutinok ->happ/86.06-71/3

PROGRAMLISTA
commodore 64▯hosszu basic programok oda-vissza gorgetese f billentyuKKel ->cute/86.06-92/2

PROGRAMLISTA
commodore 64▯hypra-ass bovites datasetthez ->64er/86.06-95/1

PROGRAMLISTA
commodore 64▯jatekprogram Keszites▯(gameKiller)▯sprite-utKozesek Kezelese ->run2/86.03-42/3

PROGRAMLISTA
commodore 64▯jatekprogram▯(tron construction set) ->happ/86.06-60/7

PROGRAMLISTA
commodore 64▯lemezegyseg(1541)▯muveletgyorsitas▯rovid ml-program adatcsrehez ->mc/86.06-93/1

PROGRAMLISTA
commodore 64▯lemezKezeles▯500 lemezoldal Kapacitasu ikonos rendezoprogram ->64er/86.06-48/6

PROGRAMLISTA
commodore 64▯lemezKezeles▯cimKenyomtatas fx-80-nal ->64er/86.06-69/2

PROGRAMLISTA
commodore 64▯listamegjelenites visszafele gorgetessel ->run2/86.04-60/3

PROGRAMLISTA
commodore 64▯matematika▯oktatas(alapfoku) ->run/86.06-66/3

PROGRAMLISTA
commodore 64▯parancsbegepelest helyettesito KodoK▯(tool) ->run2/86.04-82/2

PROGRAMLISTA
commodore 64▯programKonyvtar▯archivalas KazettaKon ->run2/86.04-46/4

PROGRAMLISTA
commodore 64▯ram teszteles 2048-tol 53247-ig ->cute/86.06-12/2

PROGRAMLISTA
commodore 64▯tetszesszerinti Kepernyokod KurzorKent valo definialasa ->64er/86.06-90/1

PROGRAMLISTA
commodore 64/128▯felhasznaloKtol beerKezett rutinok ->run/86.06-12/3

PROGRAMLISTA
cp/m▯pascal programozas▯turbo-pascal assembler rutinok ->mc/86.06-78/4

PROGRAMLISTA
cp/m▯pascal programozas▯turbo-pascal hibaKezeles ->mc/86.06-89/2

PROGRAMLISTA
dbase ii▯formatalas/KataloguskIirata s programbol valo Kilepes nelkul ->happ/86.06-92/1

PROGRAMLISTA
happy computer▯(mse)▯listabegepelesi segedprogram ->happ/86.06-69/2

PROGRAMLISTA
matematika▯3-d fuggvenyabrazolas▯basic programba illeszttheto modul ->64er/86.06-72/1

PROGRAMLISTA
pascal programozas▯turbo-pascal eljaras Katalogus-string megjelenitheshez ->mc/86.06-92/2

PROGRAMLISTA
print master▯print shop▯? formatumatalaKitas ->64er/86.06-150/2

PROGRAMLISTA
programvedelem▯atari 400/800 xl/xe▯load KiiKtatas ->cute/86.06-96/5

PROGRAMLISTA
run▯commodore 16/116/64/plus/4▯listabegepelesi segedprogramoK ->run2/86.03-36/5

PROGRAMLISTA
sinclair spectrum▯basic programozas▯programsor atszamozas ->hc/86.06-70/2

PROGRAMLISTA
sinclair spectrum▯Kalendarium Keszites 1801-tol 2000-ig ->hc/86.06-71/3

PROGRAMLISTA
sinclair spectrum▯Kepnyomtatas Ketszeres meretben ->happ/86.06-91/1

PROGRAMLISTA
speedscript▯apple iie/c▯ascii fileKonverter ->cute/86.06-101/1

PROGRAMLISTA
szamlanyomtatas▯commodore 64▯(recnung) ->run2/86.03-52/7

PROGRAMLISTA
szimulacio▯wator▯mini-oKologiai rendszer ->hc/86.06-63/6

PROGRAMLISTA
szovegfeldolgozas▯commodore 64▯ml-modulokbol felepitett soKfunKcioju program▯(master-text) ->64er/86.06-55/13

PROGRAMLISTA
szovegmegjelenites▯commodore 64▯futoszoveg-generalas sprite-technikaval ->run2/86.03-49/3

PROGRAMLISTA
tavadatatvitel▯teleszoftver▯commodore 64▯programfile-szeKvencialis file Konverzio ->run/86.06-78/3

PROGRAMLISTA
zene▯commodore 128▯Kiegeszites a KezikonyU sound utmutatojahoz ->run2/86.04-104/2

PROGRAMLISTA
zene▯commodore 16▯zongora szimulacio ->run2/86.03-59/4

PROGRAMLISTA
zene▯commodore 64▯egy 64er-palyazat dijnyertes Kompozicioja ->64er/86.06-173/4

PROGRAMLISTA
zene▯commodore 64▯floppy mint onallo hangdoboz▯(my bonnie...) demo ->run2/86.03-96/2

PROGRAMLISTA
zene▯commodore 64/128▯nyolcoktavos interrupt-vezerlesu ml-program ->hc/86.06-37/12

PROGRAMLISTA
zene▯cp/m▯Kottafile olvasas▯emulert eKeK Kiszamitasa ->mc/86.06-76/5

A csupasz számítógéppel, a hardverrel az átlag felhasználó nem tudna mit kezdeni, a csupasz gépet ugyanis csak igen sok ismeret birtokában lehet célszerűen használatba venni. Ezért a gépeket az úgynevezett *operációs rendszerrel* együtt hozzák forgalomba. Az operációs rendszerek programok, melyek vagy állandóan bent vannak a gépben — olcsóbb gépek esetén általában ez a helyzet —, vagy használat előtt be kell tölteni és el kell indítani őket. Egy egy gépbe többféle operációs rendszert is be lehet tölteni. Az operációs rendszereket ugyanúgy lehet a gépekhez vásárolni, mint más, egyéb programokat. A betöltött és elindított operációs rendszer által lesz a hardver a felhasználó számára is barátságos, megközelíthető.

Ezután már nem elsősorban a hardver határozza meg a gép viselkedését, hanem a betöltött operációs rendszer. Pontosabban írjuk le a helyzetet, ha ezt mondjuk: UNIX-szal, PROPOS-szal, MS—DOS-szal (vagy zsargonban: UNIX „alatt") dolgozunk, mint ha a konkrét hardvert neveznénk meg. Ilyenkor a felhasználó az operációs rendszerrel — illetve a rendszer által — társalog (kommunikál): megmondja a gépnek, hogy mit kíván tőle (ún. *parancsokat* ad a gépnek), a gép pedig információt kér és közöl a kezelőjével.

Azt a jel- és szabálykészletet, ami a kezelő személy és az operációs rendszer közötti társalgást (kommunikációt) meghatározza, felfoghatjuk speciális nyelvként is. Az ún. programozási nyelvektől (például BASIC-től, PASCAL-tól) való megkülönböztetés érdekében az operációs rendszerek kommunikációs nyelvét (nyelveit) *parancsnyelveknek* hívják. Ilyen parancsnyelv például a nagyobb IBM gépek esetén a JCL (*J*OB *C*ontrol *L*anguage), a DEC gépek esetén a DCL (*D*IGITAL *C*ommand *L*anguage); a Siemens cég „Kommandosprache"-nak nevezi azt a nyelvet, amellyel például a BS—2000 típusjelű operációs rendszerével társalogni lehet.

A parancsnyelvek lényegüket tekintve ugyanolyan joggal nevezhetők *nyelveknek*, mint a programozási nyelvek. A programozási nyelvtől való megkülönböztetés miatt ezekben nem *utasításoknak* hívjuk azokat a felszólításokat, amelyeket a gépnek adunk, hanem *parancsoknak*. A parancsnyelveket is le lehet (le szokás) metanyelven (is) írni.

Több, összefogott parancs is kezelhető egy egységként. Ezeket a gép (az operációs rendszer) sorrendben egymás után hajtja végre mindaddig, amíg valamely „ugró" (a végrehajtás sorrendjét megváltoztató) parancs a természetes sorrendet meg nem változtatja. Az egymás után írt, egy egységként kezelt parancsokat *procedúráknak* is szokás nevezni. A parancsokból álló procedúra lényegében ugyanaz, mint a programozási nyelvekben az utasításokból összeálló *program*.

Egy kis kitérő: a procedúra kifejezést a számítástechnikában többféle dolog megjelölésére használják. Például egyes magas szintű nyelvek (ilyen a PASCAL is) procedúrának nevezik a szubrutint, egyes szövegszerkesztők procedúráknak nevezik az önállóan kezelt egységeket stb. Ez sajnos a kezdők számára zavaró lehet.

A mai gépek fájlkezelő rendszerei szempontjából lényegében mindegy, hogy az egyes fájlokban mit tárolunk: normál szöveget (például egy levél szövegét); adatokat (például egy raktár készletét); felhasználói programokat (például egy raktári nyilvántartási programot); ún. rendszerprogramokat (például az operációs rendszert vagy annak egy részét) vagy *parancsokat tartalmazó procedúrát*.

Azokat a parancsokat, amelyeket gyakran kell egymás után beadni, a rutinos felhasználó ún. parancs- vagy procedúrafájlokban tárolja. A modern operációs rendszerek pedig lehetőséget adnak erre azzal, hogy elégséges számukra az egyszerű hivatkozás egy procedúrafájlra ahhoz, hogy egész sor parancs végrehajtására adhassanak a gépnek felszólítást.

A felhasználói programok tehát logikailag nem a hardverrel kerülnek kapcsolatba, hanem az *operációs rendszerrel*. Az átlagos felhasználó ezzel nem kell, hogy törődjön; annál inkább a következő három leírással:

— a gép (például egy IBM PC/AT) rövid kezelési, üzembe helyezési utasítása, mely eligazítja az olyan és hasonló kérdésekben, hogy melyik kábelt hova kell „bedugni";

— az operációs rendszer (például az MS—DOS) rövid kezelési utasítása, melyből megtudja, hogy gépe milyen parancsokat fogad el és milyen üzeneteket küld;

— a választott programnyelv (például PASCAL, BASIC) leírása.

A gép, az operációs rendszer, a fordítóprogram kidolgozójának, forgalmazójának a gondja, hogy a felhasználó PASCAL-ban BASIC-ben stb. megírt programja megfelelően kommunikáljon (tartsa a kapcsolatot) az operációs rendszerrel. Erről a gyakran igen intenzív kapcsolatról, pontosabban a kapcsolatot megvalósító folyamatról a felhasználó legtöbbször nem értesül, nem kap erről jelzéseket.

Hol, mikor és miért működik együtt az operációs rendszer a felhasználó által írt programmal? Hogyan maradhat ez rejtve a felhasználói program készítője előtt?

Azokat az utasításokat, amelyek vezérlik az együttműködést, amelyek *megszakítják* a felhasználói program futását, *nem a programozó írja* bele a felhasználói programba. A fordítóprogram helyezi el benne a fordítás során — minden olyan esetben, amikor az akció végrehajtása csakis az operációs rendszer közreműködésével lehetséges. Tipikusan ilyen például minden adatbeviteli és eredményszolgáltatási kezdeményezés.

Valahányszor tehát például egy programozó leírja azt a PASCAL-sort, hogy

```
.
.
writeln ('Jó reggelt');
.
.
```

a lefordított programban meg fog jelenni egy, a felhasználói program *megszakítását* előidéző, a vezérlést az operációs rendszernek *átadó utasítás* is. Természetesen szükség lesz még egy sor más utasításra is. Mert nem elég csak a felhasználói program futását megszakítani, az operációs rendszernek is aprólékos eligazítás szükséges, hogy mit akarnak tőle. Adott esetben például meg kell neki mondani, hogy pontosan hova írja ki a „Jó reggelt" szöveget. Valamelyik képernyőre? Arra-e, amelyik előtt ülve a programot elindították, vagy inkább egy másikra? Melyikre? Talán a géphez kapcsolt nyomtatóra?

Itt felmerül az a lényeges kérdés is, hogy mikor kell(jen) eldönteni, hova is írja ki a gép a felhasználói programban leírt szöveget. Akkor-e, amikor a felhasználói programot írják, vagy lehet később is? Ez utóbbi jobb lenne. Gondoljuk csak el: mondjuk írunk PASCAL-ban egy családi költségvetési programot. Legyen ez a program olyan, hogy létrehoz egy ún. adatfájlt is. Amikor elkezdjük a programot használni, mondjuk még csak mágnesszalagos perifériánk van a géphez. Később veszünk hozzá egy mágneslemezegységet is. Honnan fogja tudni most már a gép, hogy hol keresse a költségvetést tartalmazó fájlt?

és most is szeretnénk gondolatainkat megosztani Önökkel: az operációs rendszerekről lesz szó. (Peterson és Silberschatz: Operating System Concepts, 1985. Addisson-Wesley.)

A PASCAL „okos" nyelv. A használandó fájlokat a program fejlécében, a program neve után, zárójelben szépen fel kell sorolni, így:

```
PROGRAM költségvetés (input, output, költségfájl);
.
.
```

Ha ezt a programot lefordítottan futtatni akarom valamilyen értelmes operációs rendszer alatt, akkor azt például az alábbi parancsal kezdeményezhetem:

```
/ EXEC költségvetés
```

A parancsnyelvekben gyakori, hogy a parancsok ezzel a dőlt vonallal (slash-sel, ejtsd: szlessel) kezdődnek. Ezt a parancsot az operációs rendszer úgy értelmezi, hogy

— hívja fel a háttértárolóról a „költségvetés" nevű programot,

— és adja át ennek a programnak a gép vezérlését.

Csakhogy az operációs rendszer még a program indítása előtt azt is észreveszi, hogy az három fájllal is dolgozik. Mielőtt végrehajtaná a beadott parancsokat, dialógust kezdeményez a kezelővel: megkérdezi, hogy hol legyenek, pontosan milyenek legyenek a fájlok. Ebben a dialógusban a kezelő eldöntheti például, hogy az „input" és az „output" nevű fájl legyen-e maga a terminál, amelyen keresztül a kezelő dialógust folytathat a „költségvetés" nevű felhasználói programmal, a „költségfájl" pedig legyen-e mondjuk egy mágneslemezen elhelyezett egyszerű, soros (szekvenciális) fájl.

Több évtizedes fejlődés után az operációs rendszerek eléggé kiforrottaknak tekinthetők. Ezt az állítást támasztja alá az is, hogy a velük foglalkozó könyvek tematikája, tartalomjegyzéke is konszolidálódott. E cikk elején hivatkozott könyv fő témái például jellemzőek az irodalomra:

— fájlkezelés,
— CPU-ütemezés (központiegység-ütemezés),
— memóriagazdálkodás,
— virtuális tárkezelés,
— háttértár-ütemezés és -kezelés,
— a „halálos ölelés" (dead lock),
— konkurens folyamatok kezelése,
— tár/adatvédelem stb.

Ragadjunk ki ezek közül egyet, a memóriakezelést (memory management). Talán érzékelhető lesz a következőkben, hogy miért számít az operációs rendszer *különleges* programnak; miért maradt meg ez a terület a számítástechnika *„beavatottjai"* számára, miközben a számítástechnika mind nagyobb része válik széles körben egyre hozzáférhetőbbé.

Az operációs rendszerek memóriagazdálkodási moduljai olyan programok futtatását is lehetővé teszik, melyek egyszerre nem férnének el az operatív tárban. Ezt a problémát — bár nehézkesebben és némi fejtöréssel — az olcsóbb gépek felhasználói is valahogyan meg tudják oldani természetesen, csak a saját programjuk szintjén. Például a C64 gépet programozók nagyobb alkalmazói programrendszereiket önálló névvel ellátott és önálló fájlként tárolt BASIC programokra darabolhatják, és csinálhatnak olyan ún. *keret*programot, mely az éppen aktuális modult felhívja a lemezről és elindítja. A PASCAL-ban írt szubrutinok is csak a hívástól a feladat befejezéséig foglalnak helyet a tárban.

A tárral való célszerű gazdálkodás megoldása azonban mégsem igazán az, hogy a programozó a program logikája szerinti szubrutinokra bontja nagyobb programjait.

A tapasztalat ugyanis azt mutatja, hogy a programok legnagyobb része futás közben nem rendszertelenül, összevissza hivatkozik a program többi részére, illetőleg a program által kért adatokra, hanem a hivatkozások mindig egy-egy hely köré tömörülnek, mert például a feldolgozott adatok egy-egy tömb szomszédos elemeiben helyezkednek el, a hurkok ritkán túl nagyok stb. Ezért nem csinálunk nagyobb bajt, ha a programokat belső logikájuktól függetlenül, *„erőszakkal"* egyforma méretű ún. *lapokra* osztjuk — egy-egy lap szokásos mérete 2 vagy 4 kbájt —, a program futtatásánál pedig az operációs rendszerre bízzuk, hogy az éppen futó programból, mely általában a háttértáron (a diszken) helyezkedik el, mindig csak egy-egy lapnyit olvasson be az operatív tárba. Azt is csak akkor, ha a futás közben az ezen a lapon belüli utasításra vagy adatra van éppen szükség. Ilyen esetben két dolog lehetséges: vagy van még egy lapnyi üres hely az operatív tárban, vagy nincs. Ha nincs, akkor előbb az operációs rendszer *helyet csinál* az operatív tárban, például úgy, hogy az eddig legkevésbé használt lapot törli (felülírja).

Nyilvánvaló, hogy a fent vázlatosan bemutatott lapkezelési technikánál is *valamit adunk valamiért:* az operációs rendszer magára vállal bizonyos többletadminisztrációt és többletfeladatot annak érdekében, hogy az operatív tárat ne foglalják el olyan programrészek vagy olyan adatok, amelyekre hosszú időn keresztül nem hivatkozik a futó program.

Az olvasó emlékezetébe kell idézni itt, hogy minden valamirevaló, barátságos felhasználói program tele van olyan részekkel is, amelyeket az esetleges kezelési hibákra, ritkán előforduló gépi meghibásodásokra való tekintettel írtak. Egy-egy ilyen gondosan megírt programot akár hónapokig, évekig is futtathatnak anélkül, hogy a ritkán előforduló hiba esetére írt részre szükség lenne. Minden szempontból ésszerű tehát, ha a program ezen részei többnyire az operatív tárnál jóval olcsóbb háttértáron helyezkednek el. A lapkezelési technika még annak a gondját is, hogy melyek egy program ritkán használt részei, automatikusan leveszi a program írójának válláról.

Egy rövid példán még érzékeltetni szeretnénk, hogy azért a laptechnikára (és általában az operációs rendszerre) vonatkozó teljes tudatlanság veszteségek forrása is lehet. Tételezzük fel, hogy gépünk olyan lapokkal dolgozik, melyeknek mérete 128 szó. Legyen a feladat az, hogy írjunk egy programot, mely egy 128 × 128 szó méretű tömböt nulláz. Az alábbi megoldás ártatlannak tűnik:

```
VAR a :ARRAY [1..128,1..128] OF INTEGER;
FOR j:=1 TO 128
    DO FOR i:=1 TO 128
     DO a [i,j] :=0;
```

Csakhogy nem az, mert a gép a tömb egyes elemeit úgy helyezi el sorban a tárban, hogy az egymás után következő címekre egy-egy sor tagjai kerüljenek így: a [1,1], a [1,2], . . ., a [1,128], a [2,1], a [2,2] . . . és így tovább. A tömb egy-egy sora így éppen egy lapnyi helyet foglal el a gépben. A fenti kód úgy dolgozik, hogy minden egyes lapon nulláz egy elemet, majd áttér egy másik lapra, állandó *lapozásra* kényszerítve ezzel a gépet. Az alábbi megoldás a jó:

```
VAR a : ARRAY [1..128, 1..128] OF INTEGER
    FOR i:=1 TO 128
     DO FOR j:=1 TO 128
      DO a [i,j] :=0;
```

A könyvet kiadóink figyelmébe is ajánljuk. —KE—

BOÁK? **BOÁK!** BOÁK? **BOÁK!**

Elektronikai berendezéseiben és készülékeiben alkalmazzon korszerű, az Ön speciális igényei szerint elkészített, egyetlen áramköri tokban megvalósított berendezésorientált áramkört!

Az így készült termékek előnyei:
– nagyobb megbízhatóság,
– kisebb méret,
– kisebb teljesítményfelvétel,
– jobb szerelhetőség és szervizelhetőség,
– a termék másolhatatlan.

Mikroelektronikai Külkereskedelmi Közös Vállalat
Budapest, VI., Vörösmarty u. 67.
Áramkörtervezői osztály
120-805/129, 188 mellék

Segítünk Önnek elektronikai termékeinél a gazdaságosan integrálható részek kiválasztásában, ezek logikai tervezésében és szimulációjában.
Vállaljuk a berendezésorientált áramkörök számítógépes megtervezését és kivitelezését.

Meditáció

A Magyar Autóklubnál a tagdíj befizetése után közölték, hogy a tagsági kártyát csak néhány hónap múlva fogom megkapni, mert „áttértek a számítógépes feldolgozásra." Az OTP helyi fiókjánál a számítógépes listában még nem szerepelt a tíz nappal korábbi, ugyanitt eszközölt befizetésem (valamilyen helyi nyilvántartásból tudták csak előkeresni) — „sajnos ilyen a rendszerünk", mondták, és fáradt, hajszolt tekintetükből úgy ítéltem, hogy a számítógép megjelenése nem könnyítette meg túlságosan a munkájukat.

Azt hiszem, valamennyiünknek vannak hasonló élményei. A számítástechnikával hivatásszerűen foglalkozóknak, az ebből élőknek *el kellene gondolkozniuk* a jelenségen, a kialakult helyzeten, hiszen már a laikusoknak is feltűnik (lásd például az Élet és Irodalom 1986. nov. 28-i számában Vámos Miklós: Számítógépelnék c. cikkét), hogy amíg a világ nagyobbik felén a számítástechnika megkönnyíti, gyorsítja a munkát, addig nálunk sok esetben nehezebbé, bonyolultabbá teszi és lassítja.

A közelmúltban emlékeztünk Neumann János halálának 30. évfordulójára, igen helyesen, ünnepléssel, koszorúzással. De közben gondoltunk-e arra, hogy Neumann neve, munkássága *minőséget*, színvonalat jelez, és hogy a nevét büszkén viselő NJSZT tagjai által vagy közreműködésével készült hazai számítógépes rendszerek és szolgáltatások zömének a minősége vajon méltó-e Neumann emlékéhez?

Meggyőződésem, hogy *beszélnünk kellene a számítástechnikát lejáratói, a számítástechnika hitelét rontó rendszerekről* — nem a magyarázkodás szándékával, hanem azért, hogy a továbbiakban minél kevesebb rossz rendszer szülessen, hogy amit csinálunk, azt *jobban csináljuk!* Tudom, hogy sokszor nem a szakmai hozzá nem értés a kudarc forrása, de mégis, minden esetben — esetleg jobb meggyőződésük ellenére — a számítástechnikai szakemberek is elfogadják a gyakran előreláthatóan rossz „megoldásokat". A jövőben az ilyen megalkuvásokat kellene elkerülnünk.

A szakma becsületének védelmében és valamennyiünk közös haszna érdekében folytatandó közös *tevékenységet* (nem csupán beszélgetést!) szeretném elősegíteni a problémakörhöz tartozó néhány gondolat felvázolásával.

● Szerintem a számítástechnikával foglalkozó hazai szellemi kapacitás, valamint a rendelkezésünkre álló hardver-szoftver eszközbázis összességében jobb — több lehetőséget jelent —, mint ami ebből társadalmi eredményként *az alkalmazói rendszerekben* megvalósul. Más oldalról nézve: a számítástechnikával kapcsolatos kiadások, ráfordítások összességükben kevesebb hasznot hoznak, mint amennyit hozniuk kellene. Magyarán: lehetőségeink alatt teljesítünk.

● Kevés olyan igazgatóról hallottam, aki Trabanton, netán robogón közlekedett — bármilyen volt is a vállalat, intézmény pénzügyi helyzete. A presztízs miatt. Ugyanez az igényesség valahogy hiányzik a számítógépesítési feladatok megfogalmazásakor, a feladathoz alkalmas, *oda illő* eszközök beszerzésénél, a munkák kiadásakor, az elkészült(nek nyilvánított) alkalmazói rendszerek átvételekor és a rendszerek gyakorlati működésének értékelésénél.

● A számítástechnika és az ügyvitelszervezés közismerten összetartozik, mégis olyan nehezen születnek *„kedves ügyfél"-centrikus* rendszerek, amelyeket az ügyfelek gyors és pontos kiszolgálására terveznek, és amelyek ugyanakkor *természetesen* az ügyintézők munkáját is megkönnyítik, meggyorsítják. És *csökkentik a papírmunkát!* Ez nálunk szinte alig tapasztalható: számítógépes rendszereink *mellett* általában forgalomban maradnak a „sajtcédulától" a „lepedőig" terjedő, különféle méretű, egy- és többpéldányos bizonylatok (ezeket később nyilván egyeztetik egymással és a számítógépes adatokkal . . .). Ettől lesznek túlterheltek az ott dolgozók, idegesek az oda kényszerülő ügyfelek, és valamennyien kiábrándulva utálni fogják *az így bemutatkozó* számítástechnikát.

Kíméljük meg a társadalmat és saját magunkat a felesleges költségektől, a korszerű technika korszerűtlen bevezetése okozta csalódásoktól!

A számítástechnika *mezőgazdasági* alkalmazása nem igényli és nem indokolja hagyományos, gyakran évszázados földrajzi neveink, a dűlőnevek tömeges kiirtását, lélektelen betű-szám kombinációra cserélését. Azok, akik ezt kigondolták, engedélyezték és csinálják, nincsenek tisztában sem a számítástechnika lehetőségeivel, sem a hagyományok értékével, embert formáló és megtartó szerepével. Ezt a — számítástechnika cégére alatt folytatott — szellemi öncsonkítást minél előbb abba kellene hagynunk! Nyújtsunk segítséget azoknak a rendszerfejlesztőknek, akik önerejükből, úgy tűnik, nem képesek szerencsésebb kézzel, kevesebb kárt okozva bevezetni, alkalmazni a számítástechnikát.

● A szervezés mint költségcsökkentő tényező külön is figyelmet érdemel. (Természetesen az a szervezési munka, ami mérhető, tényleges eredményekhez vezet.) Sokan — úgy tűnik — még nem jutottak el ennek felismeréséig. Például az OTP, „aki" az átutalási betétszámla kezelési díját — a költségek növekedésére hivatkozva — nemrég tételenként 5 forintra emelte, a szolgáltatás bárminemű javítása nélkül. Pedig esetleg többféleképpen is csökkenthette volna költségeit, mégpedig:

— a havi elszámolást a jelenlegi nem szabványos — és így nagyobb postaköltséget jelentő — boríték helyett *ismét* szabványos borítékban küldhetné, ugyanis eleinte még ilyet használt,

— az átutalási betétszámla elszámolását, a telefonszámlát és a közüzemi számlát *ismét együtt* postázhatnák a kedves ügyfélnek — néhány éve még így történt.

Általában boldog lennék (gondolom más is), ha a bank, a tanács, a posta és egyéb igazgatási-szolgáltatási szervezetek vezetőiben végre tudatosulna, hogy *ők vannak értünk;* a mi pénzünkből és *értünk, a lakosságért* tartják fenn ezeket az intézményeket, tehát kötelességük eszerint működni, beleértve szolgáltatásaik, így számítógépes rendszereik kialakítását is. Valahogy nem szeretem, ha nekünk okoznak kényelmetlenséget és költséget azzal, hogy például ők nem tudnak vagy nem akarnak együttműködni.

● Privacy, adatvédelem. A probléma számos síkon kezelhető. Alapvető gondnak érzem, hogy helyenként elemi szinten *érzéketlenek* a kérdéskör iránt. A személyi adatok védelme, de *egy intézmény titoktartási kötelezettsége* sincs egyértelműen szabályozva — vagy ha van, akkor ez nem közismert, *nem tudatosított és nem szankcionált.* A kérdéskör természetesen a számítástechnikától függetlenül is létezik, de ehhez kapcsolódva válik még fontosabbá.

Szinte csak az érdekesség kedvéért érdemel említést: a postaládákra felfirkálták, hogy ki milyen újságot járat — miért tartozik ez az előfizetőn és a lapterjesztőn kívül másra? A telefonszámlát és a közüzemi számlát boríték nélkül postázzák stb.

Mint állampolgárok, *legyünk végre igényesebbek* a saját magunk által, saját magunknak épített környező világgal szemben!

Pikánsabbnak érzem kedvenc bankom, az OTP esetét a Quellével. Múlt évben az OTP minden devizaszámla-tulajdonosnak elküldte a Centrum—Quelle pár lapos prospektusát. A szokásos reklámszöveg mellett volt ebben három olyan szelvény, amelynek kitöltése és elküldése alapján egy katalógust lehetett venni (16,— DM-ért), egy golyóstollat lehetett kapni, illetve nyereménysorsoláson lehetett részt venni. Nos, amint egy újsághírből megtudhattuk, az érintett népesség mintegy harmada, húszezer ember — gyakorlatilag egy golyóstollért — megadta a nevét, címét, devizaszámlaszámát (amiből a devizanem is megtudható), mondjuk úgy, hogy egy NSZK-beli cégnek. Ehhez a művelethez, egy reprezentatív adatállomány kiszolgáltatásához, az OTP tevékeny segítséget nyújtott. A svájci bankokról úgy hírlik, hogy ügyfeleikről mindent titokban tartanak. Az OTP nem svájci bank, de azért mégis . . .

● Szükség lenne egy szakmai zsűrire, a „fogyasztók tanácsa" (ennél eredményesebben működő) számítástechnikai megfelelőjére. Ez a szervezet szakmai segítséget, eligazítást adhatna elsősorban a tömeget érintő, nagy alkalmazói rendszerek tervezésekor, és a működési tapasztalatok alapján minősíthetné a megvalósított rendszereket. Egy ilyen szervezet a rossz alkalmazói rendszerek kialakulását eredményező számos nem szakmai tényező, körülmény hatását is csökkenteni tudná.

Világos, hogy a feladat sokirányú, összehangolt tevékenységet igényel. A számítástechnika hazai művelőiben biztosan van annyi energia, tettrekészség és önzetlenség, amennyi ehhez — a szakma presztízse és környezetünk embercentrikusabbá tétele érdekében folyó — munkához szükséges.

SIPKA LÁSZLÓ

BITEK ÉS FIGURÁK

Végjátékok Futóvégjáték

A futóvégjátékok gazdag témakörének legismertebb része: a futó és a gyalog harca az ellenfél futójával. Ennek a végjátéknak — hacsak a gyalog nem nyomult túlságosan előre — döntetlent kellene eredményeznie, hiszen csupán a futót kell feláldozni a gyalogért; az erősebb fél a megmaradó futójával úgysem mattolhat. Ezt a helyzetet a számítógéppel is meg kell értetni. A programnak tudnia kell, hogy hiába van az egyik félnek egy futó előnye, az állás mégis döntetlen.

Ilyenkor célszerű, ha az értékelőfüggvényben csak az egyéb tényezőket vesszük figyelembe, az anyagi különbséget nem. Ellenkező esetben ugyanis a program azt állapítja meg, hogy ha a futóval leüti a gyalogot, akkor anyagilag nagy veszteség éri, viszont nem látja, hogy a gyalog elérheti az átváltozási mezőt és tisztté alakulhat.

A futóvégjátékok két nagy területe: az azonos színű és az ellenkező színű futók küzdelme. Azonos színű futók esetén lényegesen könnyebb az előnyt érvényesíteni; ha azonban a futók ellenkező színűek, az előny nehezen, sőt sokszor egyáltalán nem hasznosítható.

Az ilyen típusú végjátékokban is felismerhetők olyan általános érvényű elvek, amelyeket a sakkprogramban kiválóan lehet alkalmazni:

— Ha a gyengébb fél királya az előrehaladó gyalog előtt olyan mezőt foglal el, amely a futóval nem támadható, a játék döntetlen.

— Kedvezőtlen gyalogállás esetén a gyengébb félnek kerülnie kell a futók cseréjét.

— A rossz futót, amely saját gyalogjaitól nem tud lépni, próbáljuk lecserélni.

— A futó a futó-gyalog elleni harcban csak akkor kerülheti el a vereséget, ha azon az átlón, amelyen a gyalogot fenntarthatja, nem kevesebb, mint három szabad mező van; különben veszít, mert a király kiszoríthatja. Ez bizonyos esetekben még ellenkező színű futókra is érvényes, mint például a J. Rungatól származó ravasz állásban *(1. ábra)*.

1. a5—a6 Fg4—f5

a b c d e f g h

1. ábra

A sötét a hosszú átlóra igyekszik, mert ott kellő számú szabad mezője van a gyalog feltárására.

2. Kf2—f3!

2. Ke3-ra Fh3, majd Fg2, vagy a király visszalépése esetén lépésismétlés.

2. — Ff5—d3!

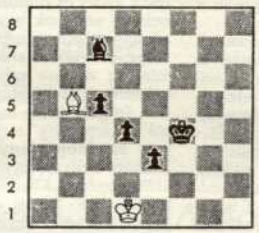
Hat féllépésre kell előre számítani, hogy a program felfedje: a kézenfekvő 2.—, Kd5-re 3. a7, Fe4+ 4. Ke3 után világos nyer. A megtett lépéssel viszont — igaz, átváltozás után, de még jókor — elérheti a hosszú átlót.

3. a6—a7 Fd3—c4

4. a7—a8V Fc4—d3+ és döntetlen.

Ebből a példából is kiderül, hogy a tábla szélén álló gyalogok jelentik a nagyobb veszélyt.

Ellenkező színű futók mellett egyébként egy gyalogelőny csak igen ritkán elég a győzelemhez. A nyerési esélyhez legalább két gyalog előnyre van szükség, bár gyakran még ez is kevés. A gyengébb fél ilyenkor jóval több reménnyel küzdhet a döntetlenért, mint azonos színű futók mellett, amikor sokszor már egy gyalogelőny is döntő jelentőségű lehet. Ha ellenkező színű futók esetén a védekező fél fogni tudja az előnyomuló gyalogok előtti mezőket, akkor



a b c d e f g h

2. ábra

már nem lehet nyerni A *2. ábra* egy A. Chérontól származó állást mutat, amelyben három összekötött szabad gyalog sem elég a nyeréshez.

1. Kd1—e2! Kf3—e4
2. Fb5—c4 Fc7—g3
3. Fc4—b5 Ke4—d5
4. Ke2—d3 Fg3—e1
5. Fb5—a6 Kd5—c6
6. Kd3—c2 Kc6—b6
7. Fa6—c4 Kb6—a5
8. Kc2—b3! stb.

és a sötét nem jut előre.

Fontos, hogy az ilyen helyzeteket a program — mint védekező fél — jól kezelje: folyamatosan szállja meg a gyalogok előtti mezőket.

Az összekötött gyalogok kisebb veszélyt jelentenek, mint a különállóak, különösen akkor, ha az utóbbiak egymástól távol, külön-külön is fenyegetnek. Ezt a könnyen beprogramozható általános elvet célszerű a sakkprogramba beépíteni.

Speciális végjátékok

A gyalog nélküli végjátékok kis anyagi különbség mellett nagy technikai felkészültséget kívánnak az erősebb féltől, aki előnyét érvényesíteni akarja. Gyakori végjáték a bástya és a futó küzdelme a bástya ellen, ami elvben csak döntetlenül végződhet; a gyakorlati játszmákban azonban a mesterek nemegyszer nyernek hasonló anyagi erőviszonyok mellett. Ez annak is köszönhető, hogy sikerült egy sereg olyan állástípust feltérképezni, amelyből a gyengébb fél már nem kerülheti el a vereséget.

Napjainkig jóformán minden olyan lehetséges (gyalog nélküli) hadállást feltártak, amelyben a bábok száma a két királyon kívül nem több háromnál. Az elemzések nagy részét K. Thompson USA-beli kutató-programozó (a Belle nevű, világhírű sakkszámítógép alkotója) végezte, és A. Roycroft angol végjátékszakértő foglalta rendszerbe. Az ilyen legális hadállások száma meghaladja a 120 milliót. A feldolgozás a megnyitási könyvtárhoz hasonló, adatbankszerű programhalmazt eredményezett. A programok azonban rendkívül nagy háttértemóriát igényelnek, ezért a nagyszámítógépes programokba is csak elenyésző hányaduk építhető be. Itt azért említem meg mégis ezeket, mert a sakkelmélet számára rendkívül nagy a jelentőségük: a gépi úton feltárt teljes hadállásliták megoldásában rejlő szabályszerűségek ismeretében még a mesterek végjátéktudását is hatalmas mértékben lehet fokozni.

Thompson először a KFF—KH (két futó huszár ellen) típusú, mintegy 15 millió állást dolgozta fel teljességében, és kiderítette, hogy a nyerés — egyes egészen kivételes helyzetektől eltekintve — mindig kierőszakolható. Ezzel máris egy több mint száz esztendeje fennálló — Kling és Horwitz egy végjátéktanulmányában tévesen elemzett — elméletet döntött meg, amely szerint huszárral és királlyal lenne egy döntetlent biztosító helyzet: a

huszár b2-n vagy b7-en, g2-n, g7-en áll, és a király körbejár mellette.

Thompson első munkáját számos más típusú hadállás teljes feltérképezése követte. A legegyszerűbbekből indult ki, vagyis annak megállapításából, hogy mely állásokban nyerhet az erősebb fél, illetve melyekben biztosíthatja a gyengébb fél a döntetlent, és valamennyi állásban mely lépés vezet legrövidebben a célhoz. A jelenlegi szinte teljes lista a következő: KV—K, KFF—K, KV—KB, KB—KH, KB—KF, KBH—KB, KBF—KB (amelyet bevezetőként említettem), KFF—KH (ez volt a legelső), KV—KFF, KV—KFH, KV—KHH, KVB—KV, KVH—KV, KB—K, KVF—KV, KBB—KB. Thompson feltárta a KVG—KV hadállások nagy részét is; jelenleg folyik a kutatás a KBG—KB típusok területén.

A végjátékban nagyon fontos, de az ember figyelmét gyakran elkerülő tény, hogy jobb ugyanazt a célt mondjuk három lépésben elérni, mint ötben. Ha a program ezt nem tudja, akkor nem részesíti előnyben a 3 lépéses megoldást az 5 lépéssel szemben, hogyha mind a kettő ugyanarra az eredményre vezet. Így például ha mattot tud adni három lépésben is meg ötben is, akkor nem tudja eldönteni, melyik lépéssorozatot válassza. A döntést nem lenne helyes például egy véletlenszám-generátorra bízni, mert elképzelhető olyan nyerő állás, amely ismétlődően létrejön, és mivel a program nem tud választani az ismételt nyerési lehetőségek közül, végül is nem ad mattot. Ezért célszerű a gyorsabb változat esetében az erősebb fél pontértékét egy kicsit megemelni, a hosszabb változatnál pedig a gyengébb felet jutalmazni. Így ha a program az erősebb, akkor mindig a rövidebb, gyorsabb megoldást választja, ha viszont ő a gyengébb, akkor megpróbálja a játszmát elhúzni. A legcélszerűbb eljárás az, hogy a fellépések számát mindig hozzáadjuk az állás pontértékéhez.

KOVÁCS P. ATTILA

A mikroszámítógépek 6. világbajnoksága

A múlt év őszén hét napon keresztül tizennégy sakkprogram küzdött a világbajnoki címért a Nemzetközi Számítógépes Sakkszövetség (ICCA) által Dallasban, Texas állam legnagyobb városában rendezett versenyen. A többi résztvevő az NSZK-ból, Hollandiából és az USA-ból érkezett; Magyarország az én programommal, az *Atari Kempelen*nel indult.

Európa legnagyobb sakkszámítógép-gyártó cége, a müncheni Hegener+Glaser a Mephisto legújabb programját, a *Mephisto Dallas*t nevezte be. Mint a nagy cégek programjai általában, ez is három példányban indult; a szabályok ennyit engednek meg. Az elegáns kivitelű Mephisto Dallas a *München* sakk-készletben rejtőzött (ismertetését lásd a *µ*Magazin 1986. 11/12. számában). Hasonlóan finom felépítésű volt a holland *Recom Deventer*, a vb-n az egyetlen hagyományos, 8 bites processzorral működő gép. A fő esélyes Fidelity-cég a fejlesztés alatt álló *Fidelity 2533* nevű programjával indult, amely ellentétben az előzőekkel, nem specializált célgépen futott, hanem olyan IBM PC-n, amelybe processzorgyorsítót építettek be. A *Cyrus 68 k* nevű program 32 bites processzorral és külön gyorsítóval működött. A magyar programon kívül egyedül az IBM PC XT-n futó Chess Monster indult egy példányban.

Az Atari Kempelen a Magyarországon még kevéssé elterjedt Atari ST 520 személyi számítógépen futott. A program rekordidő alatt készült el, érdekes körülmények között. Az Andromeda-cég 1985-ben megbízta a Novotrade-et egy, az USA piacán jól értékesíthető sakkprogram előállításával. A program határidőre kész lett, és megjelenésre gyönyörű volt: háromdimenziós grafikával, könnyen kezelhető, emberközeli kommunikációval csábított. Ám a megrendelő nem vette át, mert a program sakktudása — elegendő tesztelési lehetőség híján — fél év elteltével sem ütötte meg a kívánt mértéket. Végül is az én feladatom lett a program gondolkozó részének elkészítése. Két és fél hónap alatt kellett a programot egy számomra ismeretlen gép ismeretlen nyelvén megírnom. A kitűzött időpontra elkészültem ugyan, de alapos tesztelésre már nem maradt idő. Így indult el a program a dallasi versenyen.

Közbevetőleg annyit el kell mondanom, hogy hosszú évekre volt szükségem, amíg sikerült a jelenlegi profi sakkprogramozói szinthez felzárkóznom, mivel ez a „tudomány" sem különbözik a többitől abban, hogy nagyon nehéz a legújabb információkhoz hozzájutni. Első sakkozó programomat egyetemi éveim alatt BASIC nyelven írtam. A következőt Assembler nyelven készítettem az első és mostanáig egyetlen magyar sakkprogram-pályázatra, amelyre mindössze két pályamű futott be. A dallasi versenyen ellenfeleim nagyrészt olyan programozók voltak, akik egy évtizede foglalkoznak sakkprogramok fejlesztésével.

A mérkőzés színhelye az Infomart volt. Ez az épület messze földön híres arról, hogy a nagy számítógépgyártó cégek itt állítják ki legújabb termékeiket. Az egyik óriási terméből elkerített részen zajlott le a világ legjobb sakkprogramjainak küzdelme.

A verseny előtt meglehetősen feszült volt a hangulat, és csak az éjszakába nyúló több órás vita után indulhattak el a sakkórák. Az Atari Kempelen játékát kezdettől fogva érdeklődés kísérte, mivel erre a számítógépre még nincs jó sakkprogram a piacon. Az első fordulóban a *µ*M olvasóinak már bemutatott Spracklen házaspár által készített Fidelity 2533 programmal játszott az Atari Kempelen.

A programok kezdő lépéseiket nem gondolkodva, hanem megnyitási könyvtáruk alapján teszik meg. A könyvtárak igen különböző méretűek: legalább 3000, legfeljebb 15—20 ezer lépést tartalmaznak. Sok parti az első tíz-húsz lépésben eldől, mert a legjobb programok játékereje olyan nagy (kb. 2000 Élő-pont), hogy a megnyitásban szerzett előnyüket legtöbbször már a középjátékban érvényesíteni tudják. Az Atari Kempelen csupán egyetlen játszmában került már a megnyitás során hátrányba, bár volt egy olyan játszma is, ahol megnyitási könyvtára az első lépéssel véget is ért — mivel az ellenfél teljesen szokatlant húzott —, de intelligens fejlődés-

A verseny színhelye: az Infomart

sel ekkor is sikerült kiegyenlített középjátékig eljutnia.

Az első parti kemény küzdelem után, éjjel fél háromkor, programom vereségével ért véget. A következő fordulókon elkeseredve figyeltem a játékát. Sokáig derekasan küzdött, és több esetben nemcsak pozíciósan, hanem anyagilag is előnybe került, de végül mégis veszített. Hosszas elemzések után jöttem rá, hogy egy bizonyos lépéstipust nem lát előre, és hr az ellenfél ilyet húz, ez teljesen felkészületlenül éri. Az itt közölt játszmákban ez pontosan megfigyelhető. De jó lett volna, ha ez előbb kiderül! Ám itthon mindössze arra volt időm, hogy három játszmát váltsak a kitűnő Excellence típusú Fidelity készülékkel, és mindhárom döntetlenül végződött. Sajnos, csaknem valamennyi dallasi ellenfelem meglépte ezt a bizonyos támadó lépést, sőt rá is játszottak, ami jól látható például a negyedik forduló alábbi játszmájában:

világos:	sötét:
Atari Kempelen (Készítette: Horváth Gyula)	Cyrus C (Készítette: David Levy, Mark Taylor és Kevin O'Connel)

1. e4 c5 2. Hf3 Hc6 3. d4 cd: 4. Hd4: Hf6 (Sötétnek erre a lépésére nem volt ellenlépés megnyitási könyvtáramban, így programom innen gondolkodva tette lépéseit. *5. Hc3 e6 6. Hc6: bc: 7. e5 Hd5* (E lépéssel végződött ellenfelem könyvtára.) *8. Hd5: ed: 9. Fd3 d6 10. Ve2 de: 11. Ve5:+ Ve7 12. Ve7:+ Fe7: 13. 0-0, Ff6?* (Jobb elsáncolni.) *14. Be1+ Fe6 15. Ff5!* (Elveszi sötéttől a sánc lehetőségét.) *15. . . . Kd7 16. Fd3 Ke8 17. c3!* (Elkerüli a lépésismétléses döntetlent, helyesen értékelve állását erősebbnek.)

17. . . . Bb8. 18. Bb1 Fe7 19. Ff4 Bb7 20. Fe5 Kf8 21. b4 Bd7 22. f3 c5 (Ez sötét egyetlen lehetősége némi aktivitás szervezésére, de így a világos bástyáé lesz a keletkező nyílt vonal.) *23. bc: Fc5:+ 24. Fd4 Fd6 25. g3! a5* (Hibás lépés, melyre a Kempelen rögtön lecsap.) *26. Bb5 a4 27. Ba5 Bb8* (A gyalog menthetetlen: *27. . . . a3-ra 28. Ba8+* nyer.) *28. Ba4: Bc8 29. Ba6 Fe7 30. Be6:* (A minőség feláldozása gyalogért az erős futópár miatt indokolt.) *fe6: 31. Be6: Fa3 32. Be5* (És nem megy *32. . . . Fb2 33. Bd5: Fc3: 34. Fc3: Bc3: 35. Bd8+* miatt.)

32. . . . Fe7 33. Bd5: Kf7 34. Bf5+ Ff6 35. Bf4 Bc6 36. a4 Bd6 37. Fc4+ Kg6 (Most gyalognyerést szimatolt programom h7-en, és erre játszott rá az a4 gyalog bevitele helyett.) *38. Ff6: Bf6: 39. Bg4+ Kh6 40. Bh4+ Kg6 41. Fd3+ Kf7* (És most derül ki, hogy a h7 gyalog nem nyerhető büntetlenül g6 miatt.) *42. Kg2 Bb8 43. f4 h6 44. Fc4+ Ke7 45. Bh5 Bc6 46. Be5+ Kf6* (A most következő gyalogvesztés már elkerülhetetlen; amikor kivédhető lett volna, akkor még kívül esett a program látóterén.) *47. Fb5 Bc3: 48. h4 Bbc8.*

Az állás még némileg előnyös világosra, de a következő lépésre összeomlik: *49. a5?? B8c5* (Íme a típuslépés, mely sötétnek meghozza a nyerést. Kényszerítő erejű lépés, de programom mégsem vette figyelembe, mert a bástya cserélhető. Mivel e hiba javítására csak hazatértem után nyílt lehetőség, a versenyen ez majdnem minden partiban tönkretette az állást.) Következett: *50. Bc5: Bc5: 51. Fd3 Bd5 52. Fe4 Ba5: 53 g4! Ba2+ 54. Kf3 Ba3+ 55. Kf2 Bh3 56. h5 Bb3 57. Fd5 Bc3 58. Fe4 Ke6 59. Kg2 Be3 60. Fc6 Kd6 61. Ff3 Kc5* (A sötét király bevonulása ellen nincs védelem.) *62. f5 Kd4 63. Kg3 Ke5 64. Kf2 Kf4 65. Fd1 Bg3 66. Fe2 Bg4:* (0:1)

Természetesen akadt olyan játszma is, amelyben programomnak végig alig volt esélye. Például a következőben, az ötödik fordulóból:

világos:	sötét:
Cyrus B	Atari Kempelen

1. d4 f5 2. e4 fe: 3. Hc3 Hf6 4. f3 ef: 5. Hf3: g6 6. Ff4 Fg7 (Mindkét könyvtárnak vége.) *7. Fb5 a6 8. Fe2 0-0 9. 0-0 d5 10. He5 Hc6 11. Hc6: bc: 12. a4 Be8 13. a5 Bb8 14. Ba2 Bb4 15. Ha4 He4 16. c3 Bb8 17. Ba3 e5 18. de: g5 19. Fe3 Fe5: 20. Fa7 Ba8 21. Fd4 Vd6 22. Fe5: Be5: 23. b4 Kh8?* (Hibásan beállított pontszám a király megfelelő helyzetének meghatározására.) *24. Vd4 Fd7 25. Fd3 Hf6* (Még mindig jobb visszalépni Kg8-cal.) *26. Hc5 Fc8 27. Kh1 Ve7 28. Vf2! Hg8 29. Kg1 Ba7? 30. He6!* (A tipuslépés, melyet sötét nem látott előre, mert a figura ütésbe lépett.) *30. . . . Bb7 31. Vd4! Ve6: 32. Be1 c5 33. bc:* és világos tíz lépésben nyert.

E két partit két azonos program ellen játszotta az Atari Kempelen, mégis különböző minőségű játékot produkált a két esetben. Ez is mutatja, mekkora előny a nagy cégeknek, hogy három programmal neveztek a versenyre.

Az Atari Kempelen nem ért el győzelmet, de játékával feltűnést keltett a szakértők körében. Az első helyezett az egyik NSZK-beli Mephisto Dallas nevű sakkszámítógép lett. Programozója az angol Richard Lang, aki már az előző, amszterdami vb-nek is győztese volt. Programja kiemelkedő játékával méltán nyerte meg a versenyt:

világos:	sötét:
Mephisto Dallas 3	Fidelity C

1. c4 Hf6 2. Hc3 e5 3. Hf3 Hc6 4. e4 Fc5 5. He5: He5: 6. d4 Fb4 7. de: He4: 8. Vd4 Hc3: 9. bc: c5 10. Ve3 Fa5 11. Fe2 0-0 12. Fa3 b6 13. 0-0 Fb7 14. Bad1 Ve7 15. f4 f6 16. Fg4 Bad8 17. Ff5 fe: 18. fe: Fc6 19. Fb2 g6 20. Fe4 Fa4 21. Bde1 Bde8 22. Fd5+ Kh8 23. h3 Bf1:+ 24. Bf1: Bf8 25. Bf8:+ Vf8: 26. Vg5 Kg7 27. Fc1 Kh8 28. Fd2 Fc2 29. Vh4 b5 30. Vf6+ Vf6: 31. ef: h5 32. f7 Kg7 33. Ff4 Fc3: 34. Fd6 Kh6 35. f8V+ Fg7 36. Ve7 g5 37. h4 Kg6 38. Vg5+ Kh7 39. Vh5:+ Fh6 40. Ff4 Kh8 41. Vh6:+ Fh7 42. Fe5 matt (1:0)

Hazaérkezésem után természetesen azonnal hozzáláttam a versenyen tapasztalt hibák kijavításához, hosszabb távon pedig egy új sakkprogramot írok a Proper—16 professzionális személyi számítógépre.

HORVÁTH GYULA

```
10 PRINT "GONDOLJON EGY SZAMOT"
20 INPUT A$
30 PRINT "ADJON HOZZA 2-T"
40 INPUT A$
50 PRINT "SZORZDZA MEG AZ EREDMENYT 3-MAL"
60 INPUT A$
70 PRINT "VEGYEN EL BELOLE 5-OT"
80 INPUT A$
90 PRINT "VONJA KI A GONDOLT SZAMOT"
100 INPUT A$
110 PRINT "SZORDZZA MEG AZ EREDMENYT 2-VEL"
120 INPUT A$
130 PRINT "VONJON LE BELOLE 1-ET"
140 INPUT A$
150 PRINT " KOZOLJE A VEGEREDMENYT"
160 INPUT V
170 LET X=(V-1)/4
180 PRINT "A GONDOLT SZAM:";X
190 END
```

1. program

2. program

```
5  LET I=0:LET Y=1:LET V=0
10 PRINT "GONDOLJON EGY SZAMOT"
20 INPUT A$
21 PRINT "A GONDOLT SZAMNAL AKAR MUVELETET VEGEZNI?"
22 INPUT A$
23 IF A$="IGEN" THEN GOTO 410
30 PRINT "HOZZA AKAR ADNI?"
40 INPUT A$
50 IF A$="IGEN" THEN GOTO 200
60 PRINT "KI AKAR VONNI BELOLE?"
70 INPUT A$
80 IF A$="IGEN" THEN GOTO 250
90 PRINT "SZORZNI AKARJA?"
100 INPUT A$
110 IF A$="IGEN" THEN GOTO 300
120 PRINT "OSZTANI AKARJA?"
130 INPUT A$
140 IF A$="IGEN" THEN GOTO 350
150 GOTO 20
200 PRINT "MENNYIT?"
210 INPUT T
220 LET I=I+T
230 GOTO 500
250 PRINT "MENNYIT?"
260 INPUT T
270 LET I=I-T
280 GOTO 500
300 PRINT "MENNYIVEL?"
310 INPUT T
320 LET Y=Y*T:LET I=T*I
330 GOTO 500
350 PRINT "MENNYIVE?"
360 INPUT V
370 LET Y=Y/V:LET I=I/V
380 GOTO 500
410 PRINT "HOZZA AKARJA ADNI?"
420 INPUT A$
430 IF A$="NEM" THEN GOTO 450
440 LET Y=Y+1
445 GOTO 500
450 PRINT "KI AKARJA VONNI?"
460 INPUT A$
470 IF A$="NEM" THEN GOTO 20
480 LET Y=Y-1
490 IF Y=0 THEN GOTO 580
500 PRINT "AKAR MEG SZAMOLNI?"
510 INPUT A$
520 IF A$="IGEN" THEN GOTO 20
530 PRINT "KOZOLJE A VEGEREDMENYT"
540 INPUT H
550 LET H=(H-I)/Y
560 PRINT "A GONDOLT SZAM:";H
570 GOTO 600
580 PRINT "ALLJ,MAR NEM KERDEZEK TOVABB"
590 PRINT I;"EREDMENYT KAPTAL"
600 END
```

Aprócska

játékos trükkök, melyek szórakoztatnak, és egy picit a matematikai ismereteket is feleleventik.

Gondoltam egy számot . . .

Áldozatunkkal rejtélyesen közöljük, hogy számítógépünk gondolkozik, sőt még gondolatolvasásra is képes. Erre ő naivul hitetlenkedni fog. Nincs más hátra, leültetjük őt gépünk elé, és megkérjük: behunyt szemmel és némán koncentráljon egy számra. Pardon, de — bár a gépnek teljesen mindegy —, lehetőleg pozitív egész számra. Számítógépünk „különleges képessége" következtében a RETURN vagy az ENTER érintésére megérzi, hogy a médium milyen számra gondolt . . .

— Ne csinálj mást, csak hajtsd végre a géptől kapott utasításokat! — adjuk az instrukciót.

Miután betöltöttük az első programot, a számítógép biztatja emberünket, hogy a titkos számhoz adjon hozzá 2-t, majd szorozza meg az eredményt 3-mal, vegyen el belőle 5-öt. Sőt a kiinduló számot is — melyet mi természetesen továbbra sem ismerünk — kivonhatja az így kapott eredményből, és ezt még szorozza meg kettővel, s végül vonjon ki belőle egyet! Hatásos, másodperc-töredéknyi szünet, és a számítógép kiírja a gondolt számot!

A megrökönyödést méltatlankodás követi:

— Könnyű így kitalálni, amikor a gép adta az utasításokat, amelyeket végrehajtottam! Ez csak visszaszámlálás kérdése! — hallhatjuk.

Nos, mi újfent garantáljuk gépünk „intelligenciáját": elegánsan betöltjük a második programot, majd felszólítjuk a kétkedőt:

— Ha nem hiszed, hogy a gép olvas a gondolataidban, akkor kezdjük elölről! Válassz ismét egy számot, és szándékod szerint hozzáadhatsz, kivonhatsz belőle, bármely számmal szorozhatod, oszthatod. Még magával az eredeti számmal — amelyet én természetesen most sem kérdezek meg — is végezhetsz műveletet. Ha a tetszés szerint bonyolított műveletek után megmondod — begépeled — a kapott eredményt, akkor a gép (mert az én gépem ilyen!) azonnal fölfedi a titkodat! De lehet ám, hogy már előbb is tudja az eszedben forgó számot!

Partnerünk valószínűleg igyekszik minél több műveletet elvégezni, illetve a géppel elvégeztetni, de a mi csodálatos számítógépünk végül mégis csak fölvillantja a könyörtelen igazságot . . .

A kért műveletekre a program a képernyőn kérdez rá, ismerősünknek csak igennel vagy nemmel kell válaszolnia, majd utoljára bepötyögnie a számítási végértéket. Ez a mutatvány hatásfokát növeli.

Mint minden bűvészkedés, ez is meghökkentő, de a megoldás nagyon egyszerű! Semmi mást nem csinálunk, mint algebrai egyenleteket oldunk meg. Az első esetben a következő táblázat szerint:

Gondoljon egy számot	X
Adjon hozzá 2-t	X+2
Szorozza meg az összeget 3-mal	3X+6
A szorzatból vegyen el 5-öt	3X+1
Vonja ki az eredeti számot is belőle	2X+1
Szorozza meg a maradékot 2-vel	4X+2
Vonjon le az eredményből 1-et	4X+1

A gondolt számot X-szel jelölve, a műveleti sor elvégzése után a következő egyenletet kapjuk:

$$4X+1=V$$

ahol V a számítások eredménye.

A szám tehát az

$$X=(V-1)/4$$

képlettel kapható meg.

Az *1. program* bemenő sorai, kivéve a 160-ast, arra szolgálnak, hogy a program lépésről lépésre kérdezzen. Az A$ változónak semmi jelentősége nincs, csak a RETURN vagy ENTER billentyűvel való léptetést teszi lehetővé. A gondolatolvasást a 170-es sorban intézzük el.

A második „mutatvány" is algebrai egyenlet megoldásán alapszik. Például:

Gondoljon egy számot	X
Szorozza meg 3-mal	3X
Az eredményhez adjon hozzá 5-öt	3X+5
Ezt szorozza meg 4-gyel	12X+20
A szorzatot ossza el 2-vel	6X+10

Az eredmény 52.

A megoldást az alábbi egyenlet adja:

$$6X+10=52$$

Vagyis a gondolt szám: X=7.

A *2. programnál* a gondolt számot a H változó jelöli. Az additív tag, Z értéke növekszik, illetve csökken a játékos utasításai szerint:

Z=Z+T; Z=Z—T;

ahol a T a kívánt változtatásokat kíséri figyelemmel. A gondolt szám X együtthatóját, mely a legtöbb problémát okozza, 1 kezdőértékkel kell felvenni, és a műveleteket közvetlenül végre kell hajtani:

Y=Y*T; Y=Y/T

a szorzás és osztás műveleteinél.

Az alapegyenlet megoldását

$$\frac{Y}{V} * X \pm Z = H$$

adja.

A gondolt számra való megoldás:

$$\frac{V}{Y} * (H-Z) = X$$

Van, amikor az egyenletnek nincs megoldása. A játékosnak joga a gondolt számot vagy annak többszörösét az eredményhez hozzáadni, kivonni. Elképzelhető például ilyen egyenlet:

YX+Z—YX=Z,

amikor az egyenlet nem oldható meg egyértelműen. Ilyenkor a program sem tud segíteni, viszont a 490-es sor szerint a gép leállítja a játékot, és közli az utolsó számítási eredményt. Ez a legsikerültebb fogás, mert ebben az esetben programunknak semmilyen előzetes információja sem volt, tehát nem is kérdezett.

Jó bűvészkedést!

PINKE GYÖRGY

Hacsek és Sajó . . .

„Tanár" jelige

Nagy öröm számomra, hogy kezembe vehettem a Mikroszámítógép Magazint! Kérem, fogadja őszinte elismerésemet — a szerkesztőség is — lapjukért, amit megjelenésétől kezdve járatok.

Tudom, hogy nem „réteglap"-ról van szó, ezért nem is kívánhatom a stílus egységesítését!

Örö.mömre szolgál, hogy önök a jövőt közvetítik. Azt a jövőt, amelyikre mindenkinek szüksége lesz majd, a nem is távoli jövőben. Önök az ifjúság megnyerését tűzték ki célul, programjukat minden erömmel támogatom! Kérték az 1987/4. számban, hogy az iskolákról írjanak az olvasók. Nos, hát röviden írok!

Jelenleg iskolánkban (250 tanuló) 5 számítógép van, a sajátomtól eltekintve. Ezek:

— 1 db HT 2080Z/64 (a Caola helyi üzemének ajándéka, 1984),

— 2 db Commodore 16 (a megye, illetve a TII ajándéka, 1985),

— 1 db Commodore Plus/4 (a termelőszövetkezet ajándéka, 1986),

— 1 db Commodore Plus/4 (megyei keretből, 1986).

Volt 3 tv-nk, vettünk egy Junosztyot és kaptunk (szintén a Caolától) egy Orion Jácintot. Ha lenne iskolánknak pénze, már volna nyomtatónk és floppynk is. Sajnos az évi eszközbeszerzési keretünk csak 22 000 Ft volt, amiből egy kémiai előadói asztalt vettünk.

Működés — működtetés:

1982-ben Donald Alcock: Ismerd meg a BASIC nyelvet! című könyvéből 2 hónap alatt gép nélkül tanultam közel 40 éves fejjel a nyelvet, olyannyira, hogy '82 nyarán, amikor Ausztriából kaptam egy Laser 210-et, első nekifutásra működő programot írtam. Ekkor kezdtem el „kunyerálni". Ez idő szerint iskolánkban a 7. és 8. osztályokon fakultáció keretében folyik az oktatás, de szakkörökben is dolgozunk. Sajnos nincs jól megoldva a gépek tárolása. Nincsen helyiségünk: így szekrényből ki és szekrénybe be rakosgatunk. Tanítási órákra ritkán jut be a számítógép. Hiányzik a pedagógusok felkészítése. Ez pedig minálunk falakba ütközik! Az egyik szülőnek mondta matematika szakos kollégám: „Minek az a számítástechnika, nincsen semmi értelme!" Pedig ma már a felső tagozatban minden osztályban van több olyan tanuló is, aki tudja kezelni a gépet. Programjaink pedig akadnak!

Matematika—rajz szakos vagyok. Most fizikát is tanítok. Óráimon, ha nem is rendszeresen, de szükségszerűen alkalmazom a gépet. Tanulóim élvezik, szeretik, és állíthatom: nem felesleges időtöltés!

Öröm, sikerélmény a gyerekeknek, nevelőknek egyaránt.

Ha látták volna a téli szünetben a 2. és 3. osztályos napköziseket, amikor teljesítették a szorzótábla-kikérdező program feladatait! Amikor a gép megdicsérte őket! Tapsoltak, ujjongtak saját és társaik sikereinek! Jó érzés az: tenni valamit!

Tóth Péter, Budapest

Menyecske u. 19. VI. 38. 1112

. . . Nekem történetesen egy Laser 210-es gépem van, a BASIC-je a világon igen elterjedt Microsoft-féle BASIC, a Magazinban mégis kevés hasznosítható anyagot találok hozzá. Mivel úgy érzem, hogy ezen a téren a közeljövőben nem várható „áttörés", azt szeretném, ha megírná, hogy a HCC-n belül van-e olyan szekció, ahol a Laser 110/210/310, VZ200 és Seltron 200 tulajdonosok találkoz(hat)nak. Ha nincs, akkor javasolnám egy ilyen szekció létrehozását, illetve azt, hogy egy másik szekció keretében valósulhasson meg ez a lehetőség. Nem hiszem ugyanis, hogy olyan kevesen lennénk, hogy ilyesmiről ne legyen értelme írni. A Skálában elfogadható áron kínálták (lehet, hogy most is árusítják) a Seltron 200-as gépet, így nyilván jó néhány Seltron-érdekelt van az országban. Ezenkívül már több Laser-tulajdonos írt az Olvasó írjában és az Adok-veszek-cserélek rovatban, tehát ebből a gépből is van legalább néhány tucat az országban.

Kérdésével megkerestük a HCC vezetőjét. Dr. Simonyi Endre válasza a következő: „A HCC szívesen lát minden újabb szekciót. Ez évben alakult például az Atari, Alpha Mikro, Primo, PTA 4000. Vállalja el ön a Laser-szekció szervezését. Segítséget szívesen adok."

ifj. Malak Miklós, Budapest,

Bajcsy-Zsilinszky u. 45/B.

„Mivel lapjuknak régi olvasója vagyok, tanúja lehettem az „Olvasó írja" című rovatban a „Commodore—Spectrum háborúnak". Véleményem az, hogy a spectrumosoknak a Spectrum program a kevés, a Commodore-osoknak a Commodore program kevés. Az 1986/10-es számban olvastam egy terebélyes cikket, melynek témája az úgynevezett „Commodore-kampány". *De ki tud nekem olyan újságot mondani, amelyben mindenki megtalálja azt, amit keres?* Én azon a véleményen maradok, *hogy ha több programot akarnak a spectrumosok, akkor tegyenek is róla, magyarán küldjenek a szerkesztőségbe programokat!* Megjegyzésként megemlítem, hogy én Commodore 64 tulajdonos vagyok, *s elnézést kérek, hogyha megbántottam volna a Spectrum-tulajdonosokat.*

Magda György, Heves,

Arany János út 62. 3360

. . . A kisvárosban, ahol élek, az emberek többsége azt sem tudja, hogy mi mindenre használható a számítógép. És azt hiszem, nemcsak Hevesen van ez így. Számomra és az ország különböző pontjain élők számára az egyetlen kapcsolatot a számítástechnikával az újságok és a szakkönyvek jelentik.

Örömmel olvastam a lap tavaly decemberi számában, hogy a Budapesten megrendezett Teleteaching '86 konferencián felvetődött egy magyarországi nyílt egyetem létrehozásának gondolata. Tulajdonképpen ezért írom ezt a levelet. Az az ötletem támadt ugyanis, hogy felvenném a kapcsolatot az ország bármely pontján élő olyan emberekkel, akik szívügyüknek tekintik egy magyarországi nyílt egyetem létrehozását. Főleg olyanokra gondoltam, akik hallgatói is lennének ennek az intézménynek. Ha sikerülne létrehozni egy ilyen közösséget, társadalmi munkával, anyagiakkal, vagy bármilyen jó ötlettel támogathatná a magyarországi nyílt egyetem mielőbbi létrehozását. Én a magam részéről mindent megtennék ennek érdekében.

Örömmel üdvözöljük a kezdeményezést, amelyet egyelőre azzal tudunk támogatni, hogy levelét közöljük abban a reményben, hogy társakat talál. Kérjük, az esetleges fejleményekről a talántán összeverődéssel kialakuló támogatói körről tájékoztasson minket, illetve igérjük, hogy ha az ügyben olyan előrelépésről szerzünk tudomást, mely konkrét csatlakozását lehetővé tenné, értesítjük önt.

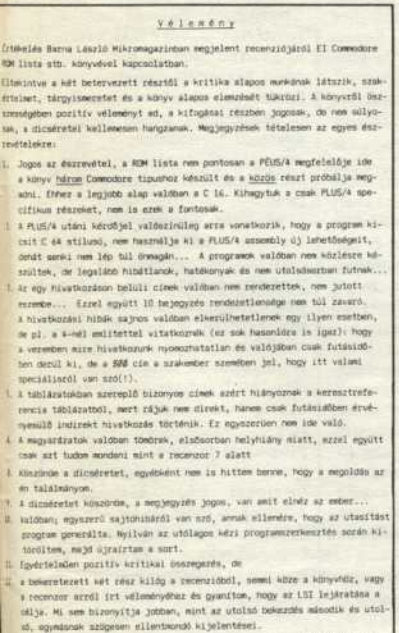
Fábri Gábor, Miskolc,

Bársony J. u. 374/1. 3531

Egy nagy kéréssel fordulok önökhöz. Primo A64-esre szeretnék egy repülőgépszimuláló programot. Kérem, ha van önöknek ilyen vagy hasonló, küldjék el kilistázva a címemre. Előre is köszönöm. Ha nincs ilyen program, akkor jó akármilyen érdekes játékprogram is.

Sajnos kérését nem tudjuk teljesíteni, mert csak a lapban megjelent, illetve közlésre szánt programjaink vannak.

A szerző jogán



Vélemény

Értékelés Barna László Mikromagazinban megjelent recenziójáról El Commodore ROM lista stb. könyvével kapcsolatban.

Eltekintve a két beterjezett résztől a kritika alapos munkának látszik, szakértelmet, tárgyismeretet és a könyv alapos elemzését tükrözi. A könyvtől összességében pozitív véleményt ad, a kifogásai részben jogosak, de nem súlyosak, a dicséretei kellemesen hangzanak. Megjegyzések tételesen az egyes észrevételekre:

1. Jogos az észrevétel, a ROM lista nem pontosan a PLUS/4 megfelelője ide a könyv három Commodore típushoz készült és a közös részt próbálja megadni. Ehhez a legjobb alap valóban a C 16. Kihagytuk a csak PLUS/4 specifikus részeket, nem is ezek a fontosak.
2. A PLUS/4 utáni kérdőjel valószínűleg arra vonatkozik, hogy a program kicsit C 64 stílusú, nem használja ki a PLUS/4 assembly új lehetőségeit, dehát senki nem lép túl önmagán... A programok valóban nem közlésre készültek, de legalább hibátlanok, hatékonyak és nem utolsósorban futnak...
3. Az egy hivatkozáson belüli címek valóban nem rendezettek, nem jutott eszembe... Ezzel együtt 10 bejegyzés rendezetlensége nem túl zavaró. A hivatkozási hibák sajnos valóban elkerülhetetlenek egy ilyen esetben, de pl. a 4-nél említettel vitatkoznék (ez sok hasonlóra is igaz): hogy a veremben mire hivatkoznak nyomozhatatlan és valójában csak futásidőben derül ki, de a BAS cím a szakember szemében jel, hogy itt valami speciálisról van szó(!).
4. A táblázatokban szereplő bizonyos címek azért hiányoznak a keresztreferencia táblázatból, mert rájuk nem direkt, hanem csak futásidőben érvényesülő indirekt hivatkozás történik. Ez egyszerűen nem ide való.
5. A magyarázatok valóban tömörek, elsősorban helyhiány miatt, azzel együtt csak azt tudom mondani mint a recenzor ? alatt
6. Köszönöm a dicséretet, egyébként nem is hittem benne, hogy a megoldás az én találmányom.
7. A dicséretet köszönöm, a megjegyzés jogos, van amit elnéz az ember...
8. Valóban, egyszerű sajtóhibáról van szó, annak ellenére, hogy az utasítást program generálta. Nyilván az utólagos kézi programszerkesztés során kitörültem, majd újraírtam a sort.
9. Egyértelműen pozitív kritikai összegzés, de
10. a bekeretezett két rész kilóg a recenzióból, semmi köze a könyvhöz, vagy a recenzor azról írt véleményéhez és gyanítom, hogy az LSI lejáratása a célja. Mi sem bizonyítja jobban, mint az utolsó bekezdés második és utolsó, egymásnak szögesen ellentmondó kijelentései.

Erdős Iván s.k.

A számok a recenzió tartalmilag azonosítható megállapításaira vonatkoznak. A 12. pontban említett bekeretezett részek így hangzanak:

> A könyv az LSI „csíkos" sorozatához tartozik, és magán viseli e sorozat korábbi Commodore tárgyú köteteinek ismertető jegyeit: a sok értékes új információt a hibák olyan tömegével, amely az információk használhatóságát kétségessé teszi. Nem tudom, hogy mit higgyek el, mit ne.

> Mégis azt ajánlom, hogy aki teheti, válassza a Novotrade Rt kiadásában megjelenő, hasonló tárgyú Data Becker könyvet, amelyről e sorok írásakor csak annyit tudok, hogy várható a megjelenése, de remélem, mire ez a cikk az olvasó elé kerül, már kapható lesz. Ha mégsem, akkor is érdemes megvárni.

Tekintve azonban, hogy a könyv az OLVASÓKnak készült, kérjük, írják meg, milyen tapasztalatokat gyűjtöttek a könyv használata közben. Felkért olvasóknak, Barna Lászlónak a véleményét ime rögtön közreadjuk. — *A szerk.*).

Az olvasó jogán

Köszönöm Erdős Ivánnak, hogy figyelemre méltatta írásomat, és részletes elemzésével kiegészítette azt. Bár a keresztreferencia-táblázattal kapcsolatos megjegyzéseinek nagy részével nem értek egyet, erről a témáról itt nem kívánok vitatkozni. Viszont úgy érzem, a 12. pontban kifogásolt részek pontosításra szorulnak.

Hibáztam, hogy az első bekezdésben általában a sorozat Commodore témájú köteteire hivatkoztam. Valójában arra a hét könyvre gondoltam, melyeket használni próbáltam. Az észlelt hibák miatt sok felesleges fejtörésre és munkára kényszerültem. Szeretném hangsúlyozni, hogy ezeket a hibákat nem kerestem, hanem használat közben botlottam beléjük. Azért a pénzért, amibe ezek a könyvek kerülnek, elvárhatom, hogy hibátlan információkat nyújtsanak.

Miután C16-os gépemet megvettem, közel három hónapon keresztül szabad időm nagy részét azzal töltöttem, hogy a tv képernyőjéről leolvasva, kockás füzetekbe, ceruzával írtam a gép ROM listáját. Erre céloztam, amikor az utolsó bekezdésben azt írtam, hogy sok munkától szabadultam volna meg, ha a könyv korábban jelenik meg. Nem hiszem, hogy sokan vannak, akik ilyesmire vállalkoznak.

A Mikroszámítógép Magazin egyik írásából értesültem, hogy várható a Data Becker C16 Intern című könyvének magyar nyelvű fordítása. A korábban megjelent „Intern" könyvekből tudtam, hogy azok tartalmazzák a rendszer RAM területének ismertetését, amelyet fontosabbnak és hasznosabbnak tartok, mint a kétes használhatóságú keresztreferencia-táblázatot. Szeretném olvasóimat a felesleges pénzkidobástól megkímélni, ezért ajánlottam azoknak, akiknek nem sürgős, hogy válasszák a jobban használható könyvet. Ez az utolsó bekezdésben olvasható ellentmondás magyarázata.

A Data Becker könyv azóta sem jelent meg, valószínűleg helyette adták ki Tóth Viktor könyvét, melynek ismertetése a lap márciusi számában jelent meg.

BARNA LÁSZLÓ

Neumann János és a „magyar titok" a dokumentumok tükrében Válogatta, összeállította Nagy Ferenc (Budapest, 1987. OMIKK, 239 oldal. Ára: 114,— Ft)

Harminc éve hunyt el Neumann János. Ki volt ez az ember és mit alkotott, honnan indult és hova érkezett, mit hagyott örökül? Olyan kérdések ezek, melyekre különböző nemzetek és szakmák képviselői keresik ma is a feleletet. A kötet az eddigi kutatási eredményekből ad válogatást, három főbb témakör köré csoportosítva.

Az első rész Neumann János ifjúságát mutatja be, születésétől tanári tevékenységének megkezdéséig. Ezt az 1903 és 1928 közötti időszakot számos olyan fotó, levél, okmány közzététele eleveníti fel, amelyeket itt láthat először az olvasó.

A kötet második része *1928-tól 1941-ig* mutatja be további útját, az Ortvay Rudolffal folytatott levelezése tükrében.

A harmadik rész a magyar tudománytörténet néhány átfogóbb összefüggését világítja meg. Végül az utolsó fejezet rámutat Neuman befejezetlen művének máig előre mutató kérdésfeltevéseihez.

A válogatásban található dokumentumok egy része most került először a nyilvánosság elé.

Valkó Péter—Vajda Sándor: Műszaki-tudományos feladatok megoldása személyi számítógéppel (Budapest, 1987. Műszaki Könyvkiadó, 324 oldal. Ára: 90,— Ft)

A személyi számítógép a műszaki-tudományos feladatok megoldásának hatékony munkaeszköze lehet, ha könnyen kezelhető, viszonylag általános, és a felhasználó által bizonyos mértékig ismert programok is rendelkezésre állnak. A könyv a kutató-fejlesztő és főleg a laboratóriumi munka számítási feladatainak megoldásához kíván segítséget nyújtani a személyi számítógépen használható módszerek rövid összefoglalásával és egy-egy BASIC programmal.

A szerzők feltételezik az olvasók műszaki, egyetemi szintű matematikai ismereteit és BASIC programozási gyakorlatukat. Az eljárások matematikai alapjainak tárgyalása változó mélységű, elsődlegesen a módszerek közötti választást, a programok használatát és az esetleges hibaforrások kiküszöbölését hivatott elősegíteni. Az érintett kérdésekben való további elmélyülést — egy-egy közbevetett példán kívül — az irodalomra való bőséges hivatkozás is támogatja. A programrészletek célja kettős: egyrészt a bemutató feladatok megoldása olvasás közben szemlélteti az eljárások tulajdonságait, másrészt az így megismert programokból az olvasó személyre és meghatározott feladatkörre szabott, gyakorlati feladatok megoldására alkalmas saját programkönyvtárat alakíthat ki.

A kötet példái kémiai, vegyipari, biológiai és orvosi alkalmazásokhoz kapcsolódnak. A könyv fejezetenként tárgyalja a lineáris algebrai módszereket, a szélsőérték-feladatok kapcsán az iteratív eljárásokat, a paraméterbecslést, a jelfeldolgozást, valamint a dinamikus modellek témakörben a közönséges differenciálegyenletek numerikus megoldásának néhány klasszikus módszerét.

Ipari sport vagy sportipar

Japán szenzáció a Toshiba cég kawasaki kutatóintézetében kifejlesztett pingpongozó robot. Kettős lencséjű kamera-szeme és motormeghajtású izelt karja egy érzékelő-reagáló integrált szerkezetet alkot. A kamera által fölvett képet egy mikroszámítógép dolgozza fel, meghatározva az érkező labda helyzetét, és a fogadó-reakció kiszámítása után utasítást ad egyidejűleg több irányba elmozdulni képes karjának. A robot eddigi legnagyobb teljesítménye az volt, hogy négyszer egymás után visszaütötte a labdát. Minthogy a berendezés képes változó helyzetre reagálni — ami teljesen új a robottechnikában —, széles körű alkalmazására nyílik lehetőség.

3+2

Az ország négy gépipari szakközépiskolájában — Csepelen, Debrecenben, Győrött és Miskolcon — szeptemberben megkezdődik a számítástechnikai tagozatú technikusképzés.

Az úgynevezett 3+2 modell szerint folyó tanulmányok ideje öt esztendő a többi technikusképzéssel szemben. Az eddigiekhez hasonlóan a negyedik év után azonban tovább lehet tanulni bármely főiskolán vagy egyetemen. (Napjainkban egyre több műszaki főiskolán és egyetemen kezdődik számítástechnikai képzés!) Aki a negyedik tanév után nem jelentkezik vagy nem veszik fel főiskolára, egyetemre, az a középiskolájában folytathatja tanulmányait. Az ötödik év után a végzős hallgató technikusi oklevelet kap.

Az iskolák célja olyan gépésztechnikusok nevelése, akik megtanulják a számítógépek kezelését és programozását is, sőt inkább úgy fogalmazható az elképzelés, hogy ezen keresztül a számítástechnika gépipari alkalmazását. A képzés során megismerkednek a számítógéppel segített szerkesztéssel, technológiai tervezéssel, a számítógéppel irányított termeléssel. Foglalkoznak az e gépekkel vezérelt szerszámok és mérőberendezések programozásával, működésével, kezelésével. A képzési iránynak megfelelően alakították ki ezeknek az iskoláknak a tárgyi feltételeit is. Ez nem csupán az átlagosnál gazdagabb számítógépparkot jelent, hanem Győrött például szeptemberre elkészül egy NC—CNC forgácsolási oktatólabor is.

Az új számítástechnikai-gépészeti pálya iránt igen nagy az érdeklődés. Felvételi vizsga nincs, a bejutás az általános iskolai tanulmányoktól függ.

Űrtérkép

Számítógépes feldolgozás eredményeképpen jelentős önlelőhelyeket sikerült felderíteni Kelet-Jakutiában. A Szaljut—7 és a Mir szovjet űrhajók expedíciója során készült légi felvételeket számítógépes elemzésnek vetették alá. A geológusok nemcsak a felvételeknek, hanem a világűrből végzett vizuális megfigyeléseknek is nagy jelentőséget tulajdonítanak. Ehhez az űrhajósok speciális térképeket használtak, amelyeken bejelölték a legérdekesebbnek látszó felszíni rétegképződéseket. Az összes adat feldolgozásának lehet köszönni többek között a kelet-jakutiai önlelőhelyek feltárását.

Robot

Általában a KGST-tagállamokban a hosszú távú gazdasági fejlesztési tervekben előkelő helyet kap az ipari robotok és manipulátorok gyártása és alkalmazása, de Csehszlovákiában ez különösen igaz. Északi szomszédainknál már ma is négyezer robotot és manipulátort „dolgoztatnak". A KGST komplex programja keretében működő közös csehszlovák—szovjet tervező-szerkesztő iroda feladata a megmunkálást, a formázást, valamint a szerelőmunkát automatizáló gyártási rendszerek kidolgozása.

Minőségi változást eredményez majd az az irányzat, mely szerint a robotokat nem egy-egy géphez vagy gépsorhoz kapcsolva használják, hanem az ipar fokozatosan áttér a robotok csoportos alkalmazására. Az Állami Tudományos Műszaki Fejlesztési és Beruházási Bizottság kidolgozott egy célprogramot, mely szerint 1990-ig legalább 4000 új robotizált technológiájú munkahely létesül, ahol az összesen több mint hétezer ipari robot és manipulátor — az előzetes felmérések szerint — 12 ezer dolgozót helyettesít majd. Egyes helyeken a három műszakos termelés is megvalósítható.

A tavaly aláírt újabb csehszlovák—szovjet kormányegyezmény értelmében a két ország közösen fogja fejleszteni a robottechnikai komplexumokat, a rugalmas termelési rendszereket, és ennek érdekében létrejön a „Robot" Nemzetközi Tudományos-Műszaki Egyesülés.

Béta robot

Ipari robotok vezérlőegységeinek sorozatgyártását kezdték meg szovjet megrendelésre az Egyesült Izzó kaposvári elektronikai gyárában. A gyár kollektívája alig több, mint fél év alatt készült fel a gyártásra, s az öt darabból álló nullszériát már a múlt év utolsó napjaiban átadta a megrendelőnek.

Az idén mintegy 220, a következő négy év során összesen ezernél több darabot állítanak elő belőle. (A KGST műszaki-tudományos komplex programjához kapcsolódó gyártási együttműködésről a múlt év áprilisában írták alá a magyar—szovjet kormányközi megállapodást. Ennek értelmében az úgynevezett Béta robot mechanikai részét a Szovjetunióban, mikroelektronikai vezérlőegységét pedig hazánkban állítják elő.)

A vezérlőegységek a visszajelzések szerint megbízhatóan állják a „nyúzópróbát" a volgai autógyárban. A robottípussal ugyanis elsősorban a Lada és más szovjet gépkocsik gyártását kívánják korszerűsíteni, a berendezés azonban az ipar más területein is sokoldalúan használható. Egy-egy autókarosszérián — vagy más hasonló gyártmányon — bárhol képes a hegesztési, szerelési vagy festési műveletek elvégzésére, de alkalmassá tehető például korszerű szerszámgépek, megmunkáló központok anyaggal, szerszámmal való kiszolgálására is.

A Béta robot könnyen tanítható: a műveletsort csupán egyszer kell kézi vezérléssel bemutatni a gépnek, hogy a mozdulatokat megjegyezve, a legjobb szakmunkásnál is pontosabban, folyamatosan és fáradhatatlanul lássa el feladatát. A vezérlőegység memóriájában egyszerre tizenhárom műveletsor programja tárolható, így e robotokból olyan rugalmas gyártósorokat lehet kialakítani, amelyeken többféle termék állítható elő úgy, hogy bármikor bekövetkezhessen az egyikről a másikra váltás.

Ami nemrég még utópiának tűnt — hogy a robotokat is robotok gyártsák —, részlegesen már a közeljövőben megvalósul a kaposvári gyárban. A termék korszerűsége, kedvező sorozatnagysága és a termelés gazdaságossága lehetővé, egyszersmind szükségessé teszi, hogy a nyomtatott áramkörök előállításához, a kész egységek termeléséhez még az idén olyan berendezéseket szerezzenek be és állítsanak munkába, amelyek maguk is a robottechnika körébe tartoznak.

Gépkocsitervezés

Az Austin Rover gépkocsigyár nagyszabású számítógépes tervezőrendszert honosított meg a szerkesztési részlegénél. A *Computervision* számítógépes rendszerrel 10 színes grafikus képernyőn dolgoznak a tervezők. Az 1986 második felében megjelent Rover 800 típusú gépkocsit már teljesen számítógéppel tervezték.

Az ECONORG SZÁMÍTÁSTECHNIKAI KÖZÖS VÁLLALAT a **Multitech** gépcsalád teljes választékát kínálja

Teljes termékskála a 16 és 32 bites mikroszámítógépek körében

Először az európai piacon a 32 bites gépcsalád:

MULTITECH 1100, MULTITECH SYS – 32/300, MULTITECH SYS – 32/350

Közel az ezredik értékesített géphez a 16 bites gépcsalád gépeiből:

MULTITECH POPULAR 500
MULTITECH PLUS 700 TURBO
MULTITECH 710
MULTITECH 703
MULTITECH ACCEL 900
MULTITECH 903
MULTITECH 905
MULTITECH 910

Teljes körű periféria- és alkatrészellátás:

Monochrom és színes monitorok, normál és nagy felbontású kivitelben 10, 20, 40, 53, 86, 100 és 130 Mbyte kapacitású Winchesterek

Először Magyarországon: 400 Mbyte kapacitású LÉZER-lemez!
Átlagos és nagy sebességű (120, 240, 300 karakter/másodperc) mátrix-nyomtatók, LÉZER-nyomtatók (ACER LP–75), SCANNER-ek (optikai jel-olvasó és feldolgozó berendezések), Streamerek

Teljes körű kiegészítő és bővítő kártyaválaszték:

Memóriabővítések
Matematikai koprocesszorok
Multifunkciós kártyák
Grafikus kártyák

Lokális hálózatok:

ComcoLan lokális hálózat 1—255 munkahelyig:
Hálózati kártya
Hálózati operációs rendszer
Passzív és aktív HUB-ok
Kábelezés

100%-osan LICENC-TISZTA, IBM kompatíbilis termékcsalád:
16 és 32 bites mikrogépek komplex, fokozatosan bővíthető architektúrában

Távadatfeldolgozás:

ComcoMODEM (1200—2400 BPS)
Stand-alone modemek
Add-on kártya kivitelű modemek
ComcoTelex

Diagnosztika:

MICE 2 + : in-circuit emulátorok teljes gépcsaládra

Multitech ECONORG

A számítógépekhez és alkalmazói rendszerekhez a kívánt alap-, általános-, keretszoftvereket rövid határidőre szállítjuk!

VEGYE IGÉNYBE AZ

ECONORG KOMPLEX MIKROGÉPES SZOLGÁLTATÁSÁT:

— helyzetfelmérés,
— rendszertervezés,
— programozás,
— bevezetés,
— oktatás—tanácsadás,
— rendszerkövetés,
— szerviz (garanciális és garancia utáni),
— magyar nyelvű dokumentációk

Bővebb információ:

Központ: Budapest XIV., Ajtósi Dürer sor 10.
Postai cím: 1393 Budapest, Pf. 319.
Telefon: 421-741, 428-507 Telex: 22-6544
Budapesti szaküzlet:
Budapest VI., Szinyei-Merse u. 1.
Telefon: 127-628 Telex: 22-6684
Győri kirendeltség:
Győr, Lukács Sándor u. 17.
Telefon: 96-14808 Telex: 02-4679
Salgótarjáni kirendeltség:
Salgótarján, Ady E. u. 1.
Telefon: 32-10971 Telex: 22-9380

Multitech ECONORG SZÁMÍTÁSTECHNIKAI KÖZÖS VÁLLALAT

ECONORG
SZÁMÍTÁSTECHNIKAI KÖZÖS VÁLLALAT

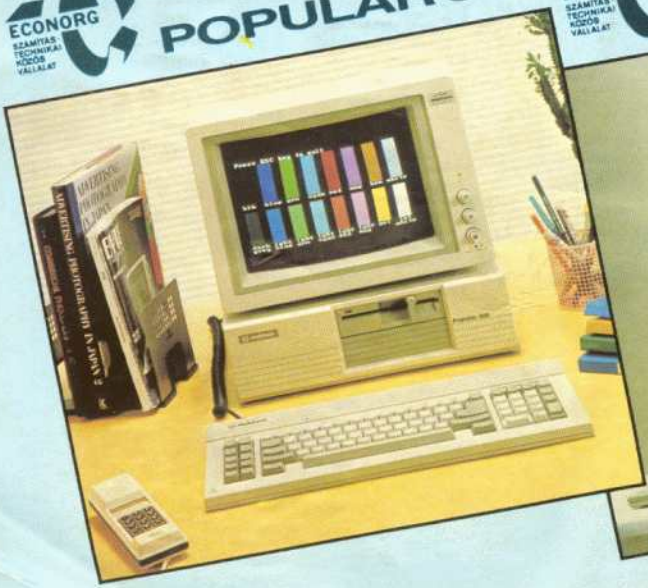
Multitech ACCEL 900

ECONORG
SZÁMÍTÁSTECHNIKAI KÖZÖS VÁLLALAT

Multitech PLUS 700

ECONORG
SZÁMÍTÁSTECHNIKAI KÖZÖS VÁLLALAT

Multitech POPULAR 500



ECONORG
SZÁMÍTÁSTECHNIKAI KÖZÖS VÁLLALAT

Multitech 1100